Unified Continuous Generative Models

Peng Sun[1,2] **Yi Jiang**[2] **Tao Lin**[1,*]
[1]Westlake University [2]Zhejiang University
sunpeng@westlake.edu.cn, yi_jiang@zju.edu.cn, lintao@westlake.edu.cn

Abstract

Recent advances in continuous generative models, encompassing multi-step processes such as diffusion and flow-matching (typically requiring 8-1000 sampling steps) and few-step methods like consistency models (typically 1-8 steps), have yielded impressive generative performance. Existing work, however, often treats these approaches as distinct learning paradigms, leading to disparate training and sampling methodologies. We propose a unified framework designed for the training, sampling, and understanding of these models. Our implementation, the **U**nified **C**ontinuous **G**enerative **M**odels **T**rainer and **S**ampler (UCGM-{T, S}), demonstrates state-of-the-art (SOTA) capabilities. For instance, on ImageNet 256×256 using a 675M diffusion transformer model, UCGM-T trains a multi-step model achieving 1.30 FID in 20 sampling steps, and a few-step model achieving 1.42 FID in 2 sampling steps. Furthermore, applying UCGM-S to a pre-trained model from prior work (1.26 FID at 250 steps) improves the FID to 1.06 using only 40 steps. Code is made publicly available at: https://github.com/LINs-lab/UCGM.

1 Introduction

(a) **NFE** $= 40$**, FID** $= 1.48$**.** (b) **NFE** $= 2$**, FID** $= 1.75$**.**

Figure 1: **Generated samples from two** 675M **diffusion transformers trained with our UCGM on ImageNet-1K** 512×512**.** The figure showcases generated samples illustrating the flexibility of Number of Function Evaluations (NFE) and superior performance achieved by our UCGM. The left subfigure presents results with NFE $= 40$ (multi-step), while the right subfigure shows results with NFE $= 2$ (few-step). Note that the samples are sampled ***without*** *classifier-free guidance or other guidance* techniques.

Continuous generative models, encompassing diffusion models [14, 35], flow-matching models [25, 29], and consistency models [37, 28], have demonstrated remarkable success in synthesizing high-fidelity data across diverse applications, including image and video generation [45, 16].

Despite their capabilities, these models often incur substantial computational costs for training and sampling [18, 20]. Furthermore, the field has evolved with these paradigms largely treated as distinct, leading to paradigm-specific training and sampling algorithms. This fragmentation presents

*Corresponding author.

Preprint. Under review.

Table 1: **Existing continuous generative paradigms as special cases of our UCGM.** Prominent continuous generative models, such as Diffusion, Flow-Matching, and Consistency models, can be formulated as specific parameterizations of our UCGM. The columns detail the required parameterizations for the transport coefficients $\alpha(\cdot), \gamma(\cdot), \hat{\alpha}(\cdot), \hat{\gamma}(\cdot)$ and parameters λ, ρ, ν of UCGM. Note that $\sigma(t)$ is defined as $e^{4(2.68t-1.59)}$ in this table.

Paradigm		UCGM-based Parameterization						
Type	e.g.,	$\alpha(t) =$	$\gamma(t) =$	$\hat{\alpha}(t) =$	$\hat{\gamma}(t) =$	$\lambda \in [0,1]$	$\rho \in [0,1]$	$\nu \in \{1,2\}$
Diffusion	EDM[18]	$\frac{\sigma(t)}{\sqrt{\sigma^2(t)+\frac{1}{4}}}$	$\frac{1}{\sqrt{\sigma^2(t)+\frac{1}{4}}}$	$\frac{-0.5}{\sqrt{\sigma^2(t)+\frac{1}{4}}}$	$\frac{2\sigma(t)}{\sqrt{\sigma^2(t)+\frac{1}{4}}}$	0	≥ 0	2
Flow Matching	OT[25]	t	$1-t$	1	-1	0	≥ 0	1
Consistency	sCM[28]	$\sin(t \cdot \frac{\pi}{2})$	$\cos(t \cdot \frac{\pi}{2})$	$\cos(t \cdot \frac{\pi}{2})$	$\sin(t \cdot \frac{-\pi}{2})$	1	1	1

two primary challenges: (a) a **lack of unified understanding**, hindering the cross-pollination of advancements between different model classes; and (b) **limited cross-paradigm generalization**, where algorithms designed for one type of model (e.g., diffusion) are often incompatible with others.

To address these limitations, we introduce UCGM, a novel framework that establishes a unified foundation for the training, sampling, and conceptual understanding of continuous generative models. Specifically, the unified trainer UCGM-T is built upon a unified training objective, parameterized by a consistency ratio $\lambda \in [0, 1]$. This allows a single training paradigm to flexibly produce models tailored for different inference regimes: models behave akin to multi-step diffusion or flow-matching approaches when λ is close to 0, and transition towards few-step consistency-like models as λ approaches 1. Moreover, this versatility can extend to compatibility with various noise schedules (e.g., linear, cosine, quadratic) without requiring bespoke algorithm modifications.

Complementing the unified trainer UCGM-T, we propose a unified sampling algorithm, UCGM-S, designed to work seamlessly with models trained via our objective. Crucially, UCGM-S is also effective for enhancing and accelerating sampling from pre-trained models developed under UCGM-T and distinct prior paradigms. The unifying nature of our UCGM is further underscored by its ability to encapsulate prominent existing continuous generative paradigms as specific instantiations of UCGM, as detailed in Tab. 1. Moreover, as illustrated in Fig. 1, models trained with UCGM can achieve excellent sample quality across a wide range of NFEs.

A key innovation within UCGM is the introduction of self-boosting techniques for both training and sampling. The training-time self-boosting mechanism enhances model quality and training efficiency, significantly reducing or eliminating the need for computationally expensive guidance techniques during inference. The sampling-time self-boosting, through methods like estimation extrapolation, markedly improves generation fidelity while minimizing NFEs. In summary, our contributions are:

(a) A unified trainer (UCGM-T) that seamlessly bridges few-step (e.g., consistency models) and multi-step (e.g., diffusion, flow-matching) generative paradigms, accommodating diverse model architectures, latent auto-encoders, and noise schedules.
(b) A versatile and unified sampler (UCGM-S) compatible with our trained models and, importantly, adaptable for accelerating and improving pre-trained models from existing distinct paradigms.
(c) A novel self-boosting mechanism integrated into the training process, which enhances model performance and training efficiency, notably reducing reliance on classifier-free guidance.
(d) An analogous self-boosting technique for the sampling process, which markedly improves generation quality while minimizing the number of function evaluations.

Extensive experiments validate the effectiveness and efficiency of UCGM. Our approach consistently matches or surpasses SOTA methods across various datasets, architectures, and resolutions, for both few-step and multi-step generation tasks (cf., the experimental results in Sec. 4).

2 Preliminaries

Given a training dataset D, let $p(\mathbf{x})$ represent its underlying data distribution, or $p(\mathbf{x}|\mathbf{c})$ under a condition $\mathbf{c}$. Continuous generative models seek to learn an estimator that gradually transforms a simple source distribution $q(\mathbf{z})$ into a complex target distribution $p(\mathbf{x})$ within a continuous space.

Typically, $q(\mathbf{z})$ is represented by the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. For instance, diffusion models generate samples by learning to reverse a noising process that gradually perturbs a data sample $\mathbf{x} \sim p(\mathbf{x})$ into a noisy version $\mathbf{x}_t = \alpha(t)\mathbf{x} + \sigma(t)\mathbf{z}$, where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Over the range $t \in [0, T]$, the perturbation intensifies with increasing t, where higher t values indicate more pronounced noise. Below, we introduce three prominent learning paradigms for deep continuous generative models.

Diffusion models [14, 38, 18]. In the widely adopted EDM method [18], the noising process is defined by setting $\alpha(t) = 1, \sigma(t) = t$. The training objective is given by $\mathbb{E}_{\mathbf{x},\mathbf{z},t}\left[\omega(t)\left\|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \mathbf{x}\right\|_2^2\right]$ where $\omega(t)$ is a weighting function. The diffusion model is parameterized by $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = c_{\text{skip}}(t)\mathbf{x}_t + c_{\text{out}}(t)\boldsymbol{F}_{\boldsymbol{\theta}}(c_{\text{in}}(t)\mathbf{x}_t, c_{\text{noise}}(t))$ where $\boldsymbol{F}_\theta$ is a neural network, and the coefficients c_{skip}, c_{out}, c_{in}, and c_{noise} are manually designed. During sampling, EDM solves the Probability Flow Ordinary Differential Equation (PF-ODE) [38]: $\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = [\mathbf{x}_t - \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)]/t$, integrated from $t = T$ to $t = 0$.

Flow matching [25]. Flow matching models are similar to diffusion models but differ in the transport process from the source to the target distribution and in the neural network training objective. The forward transport process utilizes differentiable coefficients $\alpha(t)$ and $\gamma(t)$, such that $\mathbf{x}_t = \alpha(t)\mathbf{z} + \gamma(t)\mathbf{x}$. Typically, the coefficients satisfy the boundary conditions $\alpha(1) = \gamma(0) = 1$ and $\alpha(0) = \gamma(1) = 0$. The training objective is given by $\mathbb{E}_{\mathbf{x},\mathbf{z},t}\left[\omega(t)\left\|\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - (\frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\mathbf{z} + \frac{\mathrm{d}\sigma_t}{\mathrm{d}t}\mathbf{x})\right\|_2^2\right]$. Similar to diffusion models, the reverse transport process (i.e., sampling process) begins at $t = 1$ with $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and solves the PF-ODE: $\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$, integrated from $t = 1$ to $t = 0$.

Consistency models [37, 28]. A consistency model $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ is trained to map the noisy input $\mathbf{x}_t$ directly to the corresponding clean data $\mathbf{x}$ in one or few steps by following the sampling trajectory of the PF-ODE starting from $\mathbf{x}_t$. To be valid, $\boldsymbol{f}_{\boldsymbol{\theta}}$ must satisfy the boundary condition $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, 0) \equiv \mathbf{x}$. Following EDM [18], one approach to enforce this condition is to parameterize the consistency model as $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = c_{\text{skip}}(t)\mathbf{x}_t + c_{\text{out}}(t)\boldsymbol{f}_{\boldsymbol{\theta}}\left(c_{\text{in}}(t)\mathbf{x}_t, c_{\text{noise}}(t)\right)$ with $c_{\text{skip}}(0) = 1$ and $c_{\text{out}}(0) = 0$. The training objective is defined between two adjacent time steps with a finite distance: $\mathbb{E}_{\mathbf{x}_t,t}\left[\omega(t)d\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t-\Delta t}, t - \Delta t)\right)\right]$, where $\boldsymbol{\theta}^-$ denotes $\mathrm{stopgrad}(\boldsymbol{\theta})$, $\Delta t > 0$ is the distance between adjacent time steps, and $d(\cdot, \cdot)$ is a metric function. Discrete-time consistency models are sensitive to the choice of Δt, necessitating manually designed annealing schedules [36, 11] for rapid convergence. This limitation is addressed by proposing a training objective for continuous consistency models [28], derived by taking the limit as $\Delta t \to 0$.

In summary, both diffusion and flow-matching models are multi-step frameworks operating within a continuous space, whereas consistency models are designed as few-step approaches.

3 Methodology

We first introduce our unified training objective and algorithm, UCGM-T, applicable to both few-step and multi-step models, including consistency, diffusion, and flow-matching frameworks. Additionally, we present UCGM-S, our unified sampling algorithm, which is effective across all these models.

3.1 Unifying Training Objective for Continuous Generative Models

We first propose a unified training objective for diffusion and flow-matching models, which constitute all multi-step continuous generative models. Moreover, we extend this unified objective to encompass both few-step and multi-step models.

Unified training objective for multi-step continuous generative models. We introduce a generalized training objective below that effectively trains generative models while encompassing the formulations presented in existing studies:

$$\mathcal{L}(\boldsymbol{\theta}) := \mathbb{E}_{(\mathbf{z},\mathbf{x})\sim p(\mathbf{z},\mathbf{x}),t}\left[\frac{1}{\omega(t)}\left\|\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \mathbf{z}_t\right\|_2^2\right], \tag{1}$$

where time $t \in [0,1]$, $\omega(t)$ is the weighting function for the loss, $\boldsymbol{F}_{\boldsymbol{\theta}}$ is a neural network[2] with parameters $\boldsymbol{\theta}$, $\mathbf{x}_t = \alpha(t)\mathbf{z} + \gamma(t)\mathbf{x}$, and $\mathbf{z}_t = \hat{\alpha}(t)\mathbf{z} + \hat{\gamma}(t)\mathbf{x}$. Here, $\alpha(t)$, $\gamma(t)$, $\hat{\alpha}(t)$, and $\hat{\gamma}(t)$ are unified transport coefficients over time t. Additionally, to efficiently and robustly train multi-step continuous generative models using objective (1), we propose three *necessary constraints*:

(a) $\alpha(t)$ is continuous over the interval $t \in [0,1]$, with $\alpha(0) = 0$, $\alpha(1) = 1$, and $\frac{\mathrm{d}\alpha(t)}{\mathrm{d}t} \geq 0$.
(b) $\gamma(t)$ is continuous over the interval $t \in [0,1]$, with $\gamma(0) = 1$, $\gamma(1) = 0$, and $\frac{\mathrm{d}\gamma(t)}{\mathrm{d}t} \leq 0$.
(c) For all $t \in (0,1)$, it holds that $|\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t)| > 0$ to ensure that $\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t)$ is non-zero and can serve as the denominator in (3).

Under these constraints, diffusion and flow-matching models are special cases of our unified training objective (1) with additional restrictions:

(a) For example, following EDM [18, 20], by setting $\alpha(t) = 1$ and $\sigma(t) = t$, diffusion models based on EDM can be derived from (1) provided that the constraint $\gamma(t)/\alpha(t) = t$ is satisfied[3].
(b) Similarly, flow-matching models can be derived only when $\hat{\alpha}(t) = \frac{\mathrm{d}\alpha(t)}{\mathrm{d}t}$ and $\hat{\gamma}(t) = \frac{\mathrm{d}\gamma(t)}{\mathrm{d}t}$ (see Sec. 2 for more technical details about EDM-based and flow-based models).

Unified training objective for both multi-step and few-step models. To facilitate the interpretation of our technical framework, we define two prediction functions based on model $\boldsymbol{F}_{\boldsymbol{\theta}}$ as:

$$\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_t, \mathbf{x}_t, t) := \frac{\alpha(t) \cdot \boldsymbol{F}_t - \hat{\alpha}(t) \cdot \mathbf{x}_t}{\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t)} \;\&\; \boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}_t, \mathbf{x}_t, t) := \frac{\hat{\gamma}(t) \cdot \mathbf{x}_t - \gamma(t) \cdot \boldsymbol{F}_t}{\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t)}, \tag{2}$$

where we define $\boldsymbol{F}_t := \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$. The training objective (1) can thus be transformed into:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{z},\mathbf{x}) \sim p(\mathbf{z},\mathbf{x}), t} \left[\frac{1}{\hat{\omega}(t)} \| \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \mathbf{x}_t, t) - \mathbf{x} \|_2^2 \right]. \tag{3}$$

To align with the gradient of our training objective (1), we define the new weighting function $\hat{\omega}(t)$ as $\hat{\omega}(t) := \frac{\alpha(t) \cdot \omega(t)}{\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t)}$. To unify few-step models, such as consistency models, with multi-step models, we adopt a modified version of (3) by incorporating a consistency ratio $\lambda \in [0,1]$. The loss function $\mathcal{L}(\boldsymbol{\theta})$ is then formulated as

$$\mathbb{E}_{(\mathbf{z},\mathbf{x}) \sim p(\mathbf{z},\mathbf{x}), t} \left[\frac{1}{\hat{\omega}(t)} \| \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \mathbf{x}_t, t) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_{\lambda t}, \lambda t), \mathbf{x}_{\lambda t}, \lambda t) \|_2^2 \right], \tag{4}$$

where consistency models and conventional multi-steps models are special cases within the context of (4). Specifically, setting $\lambda = 0$ yields diffusion & flow-matching models, while setting $\lambda \to 1 - \Delta t$ with $\Delta t \to 0$ recovers consistency models. Following previous studies [37], we set $\hat{\omega}(t) = \frac{\tan(t)}{4}$. The final training objective is therefore:

$$\mathbb{E}_{(\mathbf{z},\mathbf{x}) \sim p(\mathbf{z},\mathbf{x}), t} \left[\cos(t) \left\| \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t) + \frac{4\alpha(t) \Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t) \cdot (\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t))} \right\|_2^2 \right], \tag{5}$$

where the detailed derivation from (4) to (5) is provided in App. B.1.1, and we define

$$\Delta \boldsymbol{f}_t^{\mathbf{x}} := \frac{\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t), \mathbf{x}_t^\star, t) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{\lambda t}, \lambda t), \mathbf{x}_{\lambda t}^\star, \lambda t)}{t - \lambda t}. \tag{6}$$

However, optimizing the unified objective in (5) presents a challenge: stabilizing the training process as λ approaches 1. In this regime, the training dynamics resemble those of consistency models, known for unstable gradients, especially with FP16 precision [37, 28]. To address this, we propose several stabilizing training techniques stated below.

[2] For simplicity, unless otherwise specified, we assume that any conditioning information $\mathbf{c}$ is incorporated into the network input. Thus, $\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ should be understood as $\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \mathbf{c})$ when $\mathbf{c}$ is applicable.

[3] In EDM, with $\sigma(t) = t$, the input of neural network $\boldsymbol{F}_{\boldsymbol{\theta}}$ is $c_{\text{in}}(t)\mathbf{x}_t = c_{\text{in}}(t) \cdot (\mathbf{x} + t \cdot \mathbf{z})$. Although $c_{\text{in}}(t)$ can be manually adjusted, the coefficient before $\mathbf{z}$ remains t times that of $\mathbf{x}$.

Algorithm 1 (**UCGM-T**). A Unified and Efficient Trainer for Few-step and Multi-step Continuous Generative Models (including Diffusion, Flow Matching, and Consistency Models)

Require: Dataset D, transport coefficients $\{\alpha(\cdot), \gamma(\cdot), \hat{\alpha}(\cdot), \hat{\gamma}(\cdot)\}$, neural network $\boldsymbol{F}_{\boldsymbol{\theta}}$, enhancement ratio ζ, Beta distribution parameters (θ_1, θ_2), learning rate η, stop gradient operator sg.
Ensure: Trained neural network $\boldsymbol{F}_{\boldsymbol{\theta}}$ for generating samples from $p(\mathbf{x})$.

repeat
  Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{x} \sim D$, $t \sim \phi(t) := \text{Beta}(\theta_1, \theta_2)$
  Compute input data, such as $\mathbf{x}_t = \alpha(t) \cdot \mathbf{z} + \gamma(t) \cdot \mathbf{x}$ and $\mathbf{x}_{\lambda t} = \alpha(\lambda t) \cdot \mathbf{z} + \gamma(\lambda t) \cdot \mathbf{x}$
  Compute model output $\boldsymbol{F}_t = \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ and set $\mathbf{z}^\star = \mathbf{z}$ and $\mathbf{x}^\star = \mathbf{x}$
  if $\zeta \in (0, 1)$ **then**
    Let $\boldsymbol{F}_t^{\varnothing} = \boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t, \varnothing)$ to get enhanced $\mathbf{x}^\star \leftarrow \boldsymbol{\xi}(\mathbf{x}, t, \boldsymbol{f}^{\mathbf{x}}(\text{sg}(\boldsymbol{F}_t), \mathbf{x}_t, t), \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_t^{\varnothing}, \mathbf{x}_t, t))$ and $\mathbf{z}^\star \leftarrow \boldsymbol{\xi}(\mathbf{z}, t, \boldsymbol{f}^{\mathbf{z}}(\text{sg}(\boldsymbol{F}_t), \mathbf{x}_t, t), \boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}_t^{\varnothing}, \mathbf{x}_t, t))$ {Note that $\boldsymbol{\xi}(\mathbf{a}, t, \mathbf{b}, \mathbf{d}) := \mathbf{a} + \left(\zeta + \mathbf{1}_{t>s}\left(\frac{1}{2} - \zeta\right)\right) \cdot (\mathbf{b} - \mathbf{1}_{t>s} \cdot \mathbf{a} - \mathbf{d}(1 - \mathbf{1}_{t>s}))$, where $\mathbf{1}(\cdot)$ is the indicator function}
  end if
  if $\lambda \in [0, 1)$ **then**
    Compute $\mathbf{x}_t^\star = \alpha(t) \cdot \mathbf{z}^\star + \gamma(t) \cdot \mathbf{x}^\star$ and $\mathbf{x}_{\lambda t}^\star = \alpha(\lambda t) \cdot \mathbf{z}^\star + \gamma(\lambda t) \cdot \mathbf{x}^\star$
    Compute $\Delta \boldsymbol{f}_t^{\mathbf{x}} = \boldsymbol{f}^{\mathbf{x}}(\text{sg}(\boldsymbol{F}_t), \mathbf{x}_t^\star, t) \cdot (\frac{1}{t - \lambda t}) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{\lambda t}, \lambda t), \mathbf{x}_{\lambda t}^\star, \lambda t) \cdot (\frac{1}{t - \lambda t})$ {Note that for $\lambda = 0$, $\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_0, 0), \mathbf{x}_0^\star, 0) = \mathbf{x}^\star$}
  else if $\lambda = 1$ **then**
    Compute $\mathbf{x}_{t+\epsilon}^\star = \alpha(t + \epsilon) \cdot \mathbf{z}^\star + \gamma(t + \epsilon) \cdot \mathbf{x}^\star$ and $\mathbf{x}_{t-\epsilon} = \alpha(t - \epsilon) \cdot \mathbf{z}^\star + \gamma(t - \epsilon) \cdot \mathbf{x}^\star$
    Let $\Delta \boldsymbol{f}_t^{\mathbf{x}} = \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t+\epsilon}, t+\epsilon), \mathbf{x}_{t+\epsilon}^\star, t+\epsilon) \cdot (\frac{1}{2\epsilon}) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t-\epsilon}, t-\epsilon), \mathbf{x}_{t-\epsilon}^\star, t-\epsilon) \cdot (\frac{1}{2\epsilon})$
  end if
  Compute $\boldsymbol{F}_t^{\text{target}} = \text{sg}(\boldsymbol{F}_t) - \frac{4\alpha(t)}{\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t)} \cdot \frac{\text{clip}(\Delta \boldsymbol{f}_t^{\mathbf{x}}, -1, 1)}{\sin(t)}$
  Compute loss $\mathcal{L}_t(\boldsymbol{\theta}) = \cos(t) \left\| \boldsymbol{F}_t - \boldsymbol{F}_t^{\text{target}} \right\|_2^2$ and update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \int_0^1 \phi(t) \mathcal{L}_t(\boldsymbol{\theta}) \mathrm{d}t$
until Convergence

Stabilizing gradient as $\lambda \to 1$. We identify that the instability in objective (5) primarily arises from numerical computational errors in the term $\Delta \boldsymbol{f}_t^{\mathbf{x}}$, which subsequently affect the training target $\boldsymbol{F}_t^{\text{target}}$. Specifically, our theoretical analysis reveals that as $\lambda \to 1$, $\Delta \boldsymbol{f}_t^{\mathbf{x}}$ approaches $\frac{\partial \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t), \mathbf{x}_t^\star, t)}{\partial t}$. (6) then serves as a first-order difference approximation of $\frac{\partial \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t), \mathbf{x}_t^\star, t)}{\partial t}$, which is susceptible to significant computational errors. To mitigate this issue, we propose a second-order difference estimation technique by redefining $\Delta \boldsymbol{f}_t^{\mathbf{x}}$ as

$$\Delta \boldsymbol{f}_t^{\mathbf{x}} = \frac{1}{2\epsilon} \left(\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t+\epsilon}, t + \epsilon), \mathbf{x}_{t+\epsilon}^\star, t + \epsilon) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t-\epsilon}, t - \epsilon), \mathbf{x}_{t-\epsilon}^\star, t - \epsilon) \right) .$$

To further stabilize the training, we implement the following two strategies for $\Delta \boldsymbol{f}_t^{\mathbf{x}}$:

(a) We adopt a distributive reformulation of the second-difference term to prevent direct subtraction between nearly identical quantities, which can induce catastrophic cancellation, especially under limited numerical precision (e.g., FP16). Specifically, we factor out the shared scaling coefficient $\frac{1}{2\epsilon}$, namely, $\Delta \boldsymbol{f}_t^{\mathbf{x}} = \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t+\epsilon}, t+\epsilon), \mathbf{x}_{t+\epsilon}^\star, t+\epsilon) \cdot \frac{1}{2\epsilon} - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t-\epsilon}, t-\epsilon), \mathbf{x}_{t-\epsilon}^\star, t-\epsilon) \cdot \frac{1}{2\epsilon}$. In this paper, we consistently set ϵ to 0.005.
(b) We observe that applying numerical truncation [28] to $\Delta \boldsymbol{f}_t^{\mathbf{x}}$ enhances training stability. Specifically, we clip $\Delta \boldsymbol{f}_t^{\mathbf{x}}$ to the range $[-1, 1]$, which prevents abnormal numerical outliers.

Unified distribution transformation of time. Previous studies [46, 7, 37, 28, 18, 20] employ non-linear functions to transform the time variable t, initially sampled from a uniform distribution $t \sim \mathcal{U}(0, 1)$. This transformation shifts the distribution of sampled times, effectively performing importance sampling and thereby accelerating the training convergence rate. For example, the `lognorm` function $f_{\text{lognorm}}(t; \mu, \sigma) = \frac{1}{1 + \exp(-\mu - \sigma \cdot \Phi^{-1}(t))}$ is widely used [46, 7], where $\Phi^{-1}(\cdot)$ denotes the inverse Cumulative Distribution Function (CDF) of the standard normal distribution.

In this work, we demonstrate that most commonly used non-linear time transformation functions can be effectively approximated by the regularized incomplete beta function: $f_{\text{Beta}}(t; a, b) = \int_0^t \tau^{a-1}(1-\tau)^{b-1} \mathrm{d}\tau \big/ \int_0^1 \tau^{a-1}(1-\tau)^{b-1} \mathrm{d}\tau$, where a detailed analysis defers to App. B.2.1. Consequently,

Algorithm 2 (**UCGM-S**). A Unified and Efficient Sampler for Few-step and Multi-step Continuous Generative Models (including Diffusion, Flow Matching, and Consistency Models)

Require: Initial $\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, transport coefficients $\{\alpha(\cdot), \gamma(\cdot), \hat{\alpha}(\cdot), \hat{\gamma}(\cdot)\}$, trained model $\boldsymbol{F}_{\boldsymbol{\theta}}$, sampling steps N, order $\nu \in \{1, 2\}$, time schedule $\mathcal{T}$, extrapolation ratio κ, stochastic ratio ρ.
Ensure: Final generated sample $\tilde{\mathbf{x}} \sim p(\mathbf{x})$ and history samples $\{\hat{\mathbf{x}}_i\}_{i=0}^{N}$ over generation process.

Let $N \leftarrow \lfloor (N+1)/2 \rfloor$ if using second order sampling ($\nu = 2$) {Adjusts total steps to match first-order evaluation count}
for $i = 0$ to $N - 1$ **do**
  Compute model output $\boldsymbol{F} = \boldsymbol{F}_{\boldsymbol{\theta}^-}(\tilde{\mathbf{x}}, t_i)$, and then $\hat{\mathbf{x}}_i = \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}, \tilde{\mathbf{x}}, t_i)$ and $\hat{\mathbf{z}}_i = \boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}, \tilde{\mathbf{x}}, t_i)$
  if $i \geq 1$ **then**
    Compute extrapolated estimation $\hat{\mathbf{z}} = \hat{\mathbf{z}}_i + \kappa \cdot (\hat{\mathbf{z}}_i - \hat{\mathbf{z}}_{i-1})$ and $\hat{\mathbf{x}} = \hat{\mathbf{x}}_i + \kappa \cdot (\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_{i-1})$
  end if
  Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ {An example choice of ρ for performing SDE-similar sampling is: $\rho = \mathrm{clip}(\frac{|t_i - t_{i+1}| \cdot 2\alpha(t_i)}{\alpha(t_{i+1})}, 0, 1)$}
  Compute estimated next time sample $\mathbf{x}' = \alpha(t_{i+1}) \cdot (\sqrt{1-\rho} \cdot \hat{\mathbf{z}} + \sqrt{\rho} \cdot \mathbf{z}) + \gamma(t_{i+1}) \cdot \hat{\mathbf{x}}$
  if order $\nu = 2$ **and** $i < N - 1$ **then**
    Compute prediction $\boldsymbol{F}' = \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}', t_{i+1})$, $\hat{\mathbf{x}}' = \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}', \mathbf{x}', t_{i+1})$ and $\hat{\mathbf{z}}' = \boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}', \mathbf{x}', t_{i+1})$
    Compute corrected next time sample $\mathbf{x}' = \tilde{\mathbf{x}} \cdot \frac{\gamma(t_{i+1})}{\gamma(t_i)} + \left(\alpha(t_{i+1}) - \frac{\gamma(t_{i+1})\alpha(t_i)}{\gamma(t_i)}\right) \cdot \frac{\hat{\mathbf{x}} + \hat{\mathbf{x}}'}{2}$
  end if
  Reset $\tilde{\mathbf{x}} \leftarrow \mathbf{x}'$
end for

we simplify the process by directly sampling time from a Beta distribution, i.e., $t \sim \mathrm{Beta}(\theta_1, \theta_2)$, where θ_1 and θ_2 are parameters that control the shape of Beta distribution.

Learning enhanced score function. Directly employing objective (5) to train models for estimating the conditional distribution $p(\mathbf{x}|\mathbf{c})$ results in models incapable of generating realistic samples without Classifier-Free Guidance (CFG) [15]. While enhancing semantic information, CFG approximately doubles the number of function evaluations (NFE), incurring significant computational overhead.

A recent work [41] proposes modifying the target score function in [38] from $\nabla_{\mathbf{x}_t} \log(p_t(\mathbf{x}_t|\mathbf{c}))$ to an enhanced version $\nabla_{\mathbf{x}_t} \log\left(p_t(\mathbf{x}_t|\mathbf{c})\,(p_{t,\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{c})/p_{t,\boldsymbol{\theta}}(\mathbf{x}_t))^{\zeta}\right)$, where $\zeta \in (0, 1)$ denotes the enhancement ratio. This approach enables the trained models to generate highly realistic samples without relying on CFG.

Inspired by this, we propose enhancing the target score function in a manner compatible with our unified training objective (5). Specifically, we introduce a time-dependent enhancement strategy:

(a) For $t \in [0, s]$, enhance $\mathbf{x}$ and $\mathbf{z}$ by applying $\mathbf{x}^\star \leftarrow \mathbf{x} + \zeta \cdot \left(\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_t, \mathbf{x}_t, t) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_t^{\varnothing}, \mathbf{x}_t, t)\right)$, $\mathbf{z}^\star \leftarrow \mathbf{z} + \zeta \cdot \left(\boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}_t, \mathbf{x}_t, t) - \boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}_t^{\varnothing}, \mathbf{x}_t, t)\right)$. Here, $\boldsymbol{F}_t^{\varnothing} = \boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t, \varnothing)$ and $\boldsymbol{F}_t = \boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t)$.
(b) For $t \in (s, 1]$, enhance $\mathbf{x}$ and $\mathbf{z}$ by applying $\mathbf{x}^\star \leftarrow \mathbf{x} + \frac{1}{2}\left(\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_t, \mathbf{x}_t, t) - \mathbf{x}\right)$ and $\mathbf{z}^\star \leftarrow \mathbf{z} + \frac{1}{2}\left(\boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}_t, \mathbf{x}_t, t) - \mathbf{z}\right)$. We consistently set $s = 0.75$ (cf., App. B.1.4 for more analysis).

In summary, our complete algorithm UCGM-T is detailed in Alg. 1.

3.2 Unifying Sampling Process for Continuous Generative Models

In this section, we introduce our unified sampling algorithm applicable to both consistency models and diffusion/flow-based models.

For classical iterative sampling models, such as a trained flow-matching model $\boldsymbol{f}_{\boldsymbol{\theta}}$, sampling from the learned distribution $p(\mathbf{x})$ involves solving the PF-ODE [38]. This process typically uses numerical ODE solvers, such as the Euler or Runge-Kutta methods [29], to iteratively transform the initial Gaussian noise $\tilde{\mathbf{x}}$ into a sample from $p(\mathbf{x})$ by solving the ODE (i.e., $\frac{\mathrm{d}\tilde{\mathbf{x}}}{\mathrm{d}t} = \boldsymbol{f}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, t)$), where $\tilde{\mathbf{x}}$ denotes the sample at time t, and $\boldsymbol{f}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, t)$ represents the model output at that time. Similarly, sampling processes in models like EDM [18, 20] and consistency models [37] involve a comparable gradual

denoising procedure. Building on these observations and our unified trainer UCGM-T, we first propose a general iterative sampling process with two stages:

(a) Decomposition: At time t, the current input $\tilde{\mathbf{x}}_t$ is decomposed into two components: $\tilde{\mathbf{x}}_t = \alpha(t) \cdot \hat{\mathbf{z}}_t + \gamma(t) \cdot \hat{\mathbf{x}}_t$. This decomposition uses the estimation model $\boldsymbol{F}_{\boldsymbol{\theta}}$. Specifically, the model output $\boldsymbol{F}_t = \boldsymbol{F}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_t, t)$ is computed, yielding the estimated clean component $\hat{\mathbf{x}}_t = \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_t, \tilde{\mathbf{x}}_t, t)$ and the estimated noise component $\hat{\mathbf{z}}_t = \boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}_t, \tilde{\mathbf{x}}_t, t)$.
(b) Reconstruction: The next time step's input, t', is generated by combining the estimated components: $\tilde{\mathbf{x}}_{t'} = \alpha(t') \cdot \hat{\mathbf{z}}_t + \gamma(t') \cdot \hat{\mathbf{x}}_t$. The process then iterates to stage (a).

We then introduce two enhancement techniques below to optimize the sampling process:

(a) **Extrapolating the estimation.** Directly utilizing the estimated $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{z}}_t$ to reconstruct the subsequent input $\tilde{\mathbf{x}}_{t'}$ can result in significant estimation errors, as the estimation model $\boldsymbol{F}_{\boldsymbol{\theta}}$ does not perfectly align with the target function $\boldsymbol{F}^{\text{target}}$ for solving the PF-ODE.
Note that CFG [15] guides a conditional model using an unconditional model, namely, $\boldsymbol{f}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, t) = \boldsymbol{f}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, t) + \kappa \cdot \big(\boldsymbol{f}^{\varnothing}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, t) - \boldsymbol{f}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, t)\big)$, where κ is the guidance ratio. This approach can be interpreted as leveraging a less accurate estimation to guide a more accurate one [19].
Extending this insight, we propose to extrapolate the next time-step estimates $\hat{\mathbf{x}}_{t'}$ and $\hat{\mathbf{z}}_{t'}$ using the previous estimates $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{z}}_t$, formulated as: $\hat{\mathbf{x}}_{t'} \leftarrow \hat{\mathbf{x}}_{t'} + \kappa \cdot (\hat{\mathbf{x}}_{t'} - \hat{\mathbf{x}}_t)$ and $\hat{\mathbf{z}}_{t'} \leftarrow \hat{\mathbf{z}}_{t'} + \kappa \cdot (\hat{\mathbf{z}}_{t'} - \hat{\mathbf{z}}_t)$, where $\kappa \in [0, 1]$ is the extrapolation ratio. This extrapolation process significantly enhances sampling quality and reduces the number of sampling steps. Notably, this technique is compatible with CFG and does not introduce additional computational overhead (see Sec. 4.2 for details).
(b) **Incorporating stochasticity.** During the aforementioned sampling process, the input $\tilde{\mathbf{x}}_t$ is deterministic, potentially limiting the diversity of generated samples. To mitigate this, we introduce a stochastic term ρ to $\tilde{\mathbf{x}}_t$, defined as: $\tilde{\mathbf{x}}_{t'} = \alpha(t') \cdot \big(\sqrt{1-\rho} \cdot \hat{\mathbf{z}}_t + \sqrt{\rho} \cdot \mathbf{z}\big) + \gamma(t') \cdot \hat{\mathbf{x}}_t$, where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a random noise vector, and ρ is the stochasticity ratio. This stochastic term acts as a random perturbation to $\tilde{\mathbf{x}}_t$, thereby enhancing the diversity of generated samples.

Empirical results demonstrate that setting $\rho = \lambda$ consistently yields optimal performance in terms of generation quality. The theoretical analysis of this phenomenon is deferred to future work. Furthermore, empirical investigation of κ indicates that the range $[0.2, 0.6]$ is consistently beneficial (cf., Sec. 4.4). Performance is observed to be relatively insensitive within this specific range.

Unifed sampling algorithm UCGM-S. Putting all these factors together, here we introduce a unified sampling algorithm applicable to consistency models and diffusion & flow-based models, as presented in Alg. 2. This framework demonstrates that classical samplers, such as the Euler sampler utilized for flow-matching models [29], constitute a special case of our UCGM-S (cf. App. B.1.6 for analysis). Extensive experiments demonstrate two key features of this algorithm:

(a) Reduced computational resources: It decreases the number of sampling steps required by existing models while maintaining or enhancing performance.
(b) High compatibility: It is compatible with existing models, irrespective of their training objectives or noise schedules, without necessitating modifications to model architectures or tuning.

For detailed justifications of these features, please refer to Sec. 4.

4 Experiment

This section details the experimental setup and evaluation of our proposed methodology, UCGM-{T, S}. Our approach relies on specific parameterizations of the transport coefficients $\alpha(\cdot)$, $\gamma(\cdot)$, $\hat{\alpha}(\cdot)$, and $\hat{\gamma}(\cdot)$, as detailed in Alg. 1 and Alg. 2. Therefore, Tab. 2 provides a summary of the parameterizations used in our experiments. We evaluate both the training (UCGM-T) and sampling (UCGM-S) capabilities of our framework. Crucially, we also demonstrate the compatibility of UCGM-S with models trained using prior methods, in which we show how these methods/models can be represented within our unified transport coefficient framework.

4.1 Experimental Setting

The experimental settings are detailed below (additional details are provided in App. A.1).

Table 2: **Comparison of different transport types employed during the sampling and training phases of our UCGM-{T,S}.** "TrigLinear" and "Random" are introduced herein specifically for ablation studies. "TrigLinear" is constructed by combining the transport coefficients of "Linear" and "TrigFlow". "Random" represents a randomly designed transport type used to demonstrate the generality of our UCGM. Other transport types are adapted from existing methods and transformed into the transport coefficient representation used by UCGM.

	Linear	ReLinear	TrigFlow	EDM ($\sigma(t)=e^{4\cdot(2.68t-1.59)}$)	TrigLinear	Random
$\alpha(t)$	t	$1-t$	$\sin(t\cdot\frac{\pi}{2})$	$\sigma(t)/\sqrt{\sigma^2(t)+0.25}$	$\sin(t\cdot\frac{\pi}{2})$	$\sin(t\cdot\frac{\pi}{2})$
$\gamma(t)$	$1-t$	t	$\cos(t\cdot\frac{\pi}{2})$	$1/\sqrt{\sigma^2(t)+0.25}$	$\cos(t\cdot\frac{\pi}{2})$	$1-t$
$\hat{\alpha}(t)$	1	-1	$\cos(t\cdot\frac{\pi}{2})$	$-0.5/\sqrt{\sigma^2(t)+0.25}$	1	1
$\hat{\gamma}(t)$	-1	1	$-\sin(t\cdot\frac{\pi}{2})$	$2\sigma(t)/\sqrt{\sigma^2(t)+0.25}$	-1	$-1-e^{-5t}$
e.g.,	[29, 48]	[46, 44]	[28, 3]	[37, 20, 18]	N/A	N/A

Datasets. We utilize ImageNet-1K [5] at resolutions of 512×512 and 256×256 as our primary datasets, following prior studies [20, 37] and adhering to ADM's data preprocessing protocols [6]. Additionally, CIFAR-10 [22] at a resolution of 32×32 is employed for ablation studies.

For both 512×512 and 256×256 images, experiments are conducted using latent space generative modeling in line with previous works. Specifically:

(a) For 256×256 images, we employ multiple widely-used auto-encoders, including SD-VAE [32], VA-VAE [46], and E2E-VAE [23].
(b) For 512×512 images, a DC-AE (*f32c32*) [4] with a higher compression rate is used to conserve computational resources. When utilizing SD-VAE for 512×512 images, a $2\times$ larger patch size is applied to maintain computational parity with the 256×256 setting.

Consequently, the computational burden for generating images at both 512×512 and 256×256 resolutions remains comparable across our trained models[4]. Further details on datasets and auto-encoder parameters are provided in App. A.1.1.

Neural network architectures. We evaluate UCGM-S sampling using models trained with established methodologies. These models employ various architectures from two prevalent families commonly used in continuous generative models:

(a) Diffusion Transformers, including variants such as DiT [31], UViT [1], SiT [29], Lightening-DiT [46], and DDT [44].
(b) UNet-based convolutional networks, including improved UNets [18, 38] and EDM2-UNets [20].

For training models specifically for UCGM-T, we consistently utilize DiT as the backbone architecture. We train models of various sizes (B: 130M, L: 458M, XL: 675M parameters) and patch sizes. Notation such as XL/2 denotes the XL model with a patch size of 2. Following prior work [46, 44], minor architectural modifications are applied to enhance training stability (details in App. A.1.2).

Baselines. We compare our approach against several SOTA continuous and discrete generative models representative of distinct methodologies. We broadly categorize these baselines by their generation process:

(a) Multi-step models. These methods typically synthesize data through a sequence of steps. We include various diffusion models, encompassing classical formulations like DDPM and score-based models [35, 14], and advanced variants focusing on improved sampling or performance in latent spaces [6, 18, 31, 50, 1]. We also consider flow-matching models [25], which leverage continuous normalizing flows and demonstrate favorable training properties, along with subsequent scaling efforts [29, 48, 46]. Additionally, we include autoregressive models [24, 42, 47], which generate data sequentially, often in discrete domains.
(b) Few-step models. These models are designed for efficient, often single-step or few-step, generation. This category includes generative adversarial networks [12], which achieve efficient one-step synthesis through adversarial training, and their large-scale variants [2, 33, 17]. We also

[4]Previous works often employed the same auto-encoders and patch sizes for both resolutions, resulting in higher computational costs for generating 512×512 images.

Table 3: **Sample quality comparison for multi-step generation task on class-conditional ImageNet-1K.** Notation A⊕B denotes the result obtained by combining methods A and B. ↓/↑ indicate a decrease/increase, respectively, in the metric compared to the baseline performance of the pre-trained models.

512 × 512					256 × 256				
METHOD	**NFE (↓)**	**FID (↓)**	**#Params**	**#Epochs**	**METHOD**	**NFE (↓)**	**FID (↓)**	**#Params**	**#Epochs**
Diffusion & Flow-Matching Models									
ADM-G [6]	250×2	7.72	559M	388	ADM-G [6]	250×2	4.59	559M	396
U-ViT-H/4 [1]	50×2	4.05	501M	400	U-ViT-H/2 [1]	50×2	2.29	501M	400
DiT-XL/2 [31]	250×2	3.04	675M	600	DiT-XL/2 [31]	250×2	2.27	675M	1400
SiT-XL/2 [29]	250×2	2.62	675M	600	SiT-XL/2 [29]	250×2	2.06	675M	1400
MaskDiT [50]	79×2	2.50	736M	-	MDT [10]	250×2	1.79	675M	1300
EDM2-S [20]	63	2.56	280M	1678	REPA-XL/2 [48]	250×2	1.96	675M	200
EDM2-L [20]	63	2.06	778M	1476	REPA-XL/2 [48]	250×2	1.42	675M	800
EDM2-XXL [20]	63	1.91	1.5B	734	Light.DiT [46]	250×2	2.11	675M	64
DiT-XL/1⊕[4]	250×2	2.41	675M	400	Light.DiT [46]	250×2	1.35	675M	800
U-ViT-H/1⊕[4]	30×2	2.53	501M	400	DDT-XL/2 [44]	250×2	1.31	675M	256
REPA-XL/2 [48]	250×2	2.08	675M	200	DDT-XL/2 [44]	250×2	1.26	675M	400
DDT-XL/2 [44]	250×2	**1.28**	675M	-	REPA-E-XL [23]	250×2	**1.26**	675M	800
GANs & Masked Models									
VQGAN⊕[8]	256	18.65	227M	-	VQGAN⊕[40]	-	2.18	3.1B	300
MAGVIT-v2 [47]	64×2	1.91	307M	1080	MAR-L [24]	256×2	1.78	479M	800
MAR-L [24]	256×2	**1.73**	479M	800	MAR-H [24]	256×2	**1.55**	943M	800
VAR-*d*36-s [42]	10×2	2.63	2.3B	350	VAR-*d*30-re [42]	10×2	1.73	2.0B	350
Ours: UCGM-S sampling with models trained by prior works									
UCGM-S⊕[20]	$40^{\downarrow 23}$	$2.53^{\downarrow 0.03}$	280M	-	UCGM-S⊕[44]	$100^{\downarrow 400}$	$1.27^{\uparrow 0.01}$	675M	-
UCGM-S⊕[20]	$50^{\downarrow 13}$	$2.04^{\downarrow 0.02}$	778M	-	UCGM-S⊕[46]	$100^{\downarrow 400}$	$1.21^{\downarrow 0.14}$	675M	-
UCGM-S⊕[20]	$40^{\downarrow 23}$	$1.88^{\downarrow 0.03}$	1.5B	-	UCGM-S⊕[23]	$80^{\downarrow 420}$	$\mathbf{1.06}^{\downarrow 0.20}$	675M	-
UCGM-S⊕[44]	$200^{\downarrow 300}$	$\mathbf{1.25}^{\downarrow 0.03}$	675M	-	UCGM-S⊕[23]	$20^{\downarrow 480}$	$2.00^{\uparrow 0.74}$	675M	-
Ours: models trained and sampled using UCGM-{T,S} (setting $\lambda = 0$)									
⊕DC-AE [4]	40	**1.48**	675M	800	⊕SD-VAE [32]	60	1.41	675M	400
⊕DC-AE [4]	20	1.68	675M	800	⊕VA-VAE [46]	60	1.21	675M	400
⊕SD-VAE [32]	40	1.67	675M	320	⊕E2E-VAE [23]	40	**1.21**	675M	800
⊕SD-VAE [32]	20	1.80	675M	320	⊕E2E-VAE [23]	20	1.30	675M	800

evaluate consistency models [37], proposed for high-quality generation adaptable to few sampling steps, and subsequent techniques aimed at improving their stability and scalability [36, 28, 51].

Consistent with prior work [38, 14, 25, 2], we adopt standard evaluation protocols. The primary metric for assessing image quality is the Fréchet Inception Distance (FID) [13], calculated on 50,000 generated images (FID-50K). Additional details regarding evaluation metrics and baselines are provided in App. A.

Implementation details. Our implementation is developed in PyTorch [30]. Training employs AdamW [27] for multi-step sampling models. For few-step sampling models, RAdam [26] is used to improve training stability. Consistent with standard practice in generative modeling [48, 29], an exponential moving average (EMA) of model weights is maintained throughout training using a decay rate of 0.9999. All reported results utilize the EMA model. Comprehensive hyperparameters and additional implementation details are provided in App. A.

4.2 Comparison with SOTA Methods for Multi-Step Generation

Our experiments on ImageNet-1K at 512×512 and 256×256 resolutions systematically validate the three key advantages of UCGM: (1) sampling acceleration via UCGM-S on pre-trained models, (2) ultra-efficient generation with joint UCGM-T+UCGM-S, and (3) broad compatibility.

UCGM-S: Plug-and-play sampling acceleration. UCGM-S provides a method for accelerating sampling from pre-trained generative models by reducing the required number of Number of Function Evaluations (NFEs) while preserving or improving generation quality, as measured by FID. Applied to 512×512 image generation, the approach demonstrates notable efficiency gains:

(a) For the diffusion-based models, such as a pre-trained EDM2-XXL model, UCGM-S reduced NFEs from 63 to 40 (a 36.5% reduction), concurrently improving FID from 1.91 to 1.88.

Table 4: **Sample quality comparison for few-step generation task on class-conditional ImageNet-1K.**

512×512					256×256				
METHOD	**NFE (↓)**	**FID (↓)**	**#Params**	**#Epochs**	**METHOD**	**NFE (↓)**	**FID (↓)**	**#Params**	**#Epochs**
Consistency Training & Distillation									
sCT-M [28]	1	5.84	498M	1837	iCT [36]	2	20.3	675M	-
	2	5.53	498M	1837	Shortcut-XL/2 [9]	1	10.6	676M	250
sCT-L [28]	1	5.15	778M	1274		4	7.80	676M	250
	2	4.65	778M	1274		128	3.80	676M	250
sCT-XXL [28]	1	4.29	1.5B	762	IMM-XL/2 [51]	1×2	7.77	675M	3840
	2	3.76	1.5B	762		2×2	5.33	675M	3840
sCD-M [28]	1	2.75	498M	1997		4×2	3.66	675M	3840
	2	2.26	498M	1997		8×2	2.77	675M	3840
sCD-L [28]	1	2.55	778M	1434	IMM ($\omega = 1.5$)	1×2	8.05	675M	3840
	2	2.04	778M	1434		2×2	3.99	675M	3840
sCD-XXL [28]	1	2.28	1.5B	921		4×2	2.51	675M	3840
	2	**1.88**	1.5B	921		8×2	**1.99**	675M	3840
GANs & Masked Models									
BigGAN [2]	1	8.63	160M	-	BigGAN [2]	1	6.95	112M	-
StyleGAN [33]	1×2	**2.41**	168M	-	GigaGAN [17]	1	3.45	569M	-
MAGVIT-v2 [47]	64×2	1.91	307M	1080	StyleGAN [33]	1×2	**2.30**	166M	-
VAR-*d*36-s [42]	10×2	2.63	2.3B	350	VAR-*d*30-re [42]	10×2	1.73	2.0B	350
Ours: models trained and sampled using UCGM-{T, S} (setting $\lambda = 0$)									
⊕DC-AE [4]	32	**1.55**	675M	800	⊕VA-VAE [46]	16	2.11	675M	400
⊕DC-AE [4]	16	1.81	675M	800	⊕VA-VAE [46]	8	6.09	675M	400
⊕DC-AE [4]	8	3.07	675M	800	⊕E2E-VAE [23]	16	**1.40**	675M	800
⊕DC-AE [4]	4	74.0	675M	800	⊕E2E-VAE [23]	8	2.68	675M	800
Ours: models trained and sampled using UCGM-{T, S} (setting $\lambda = 1$)									
⊕DC-AE [4]	1	2.42	675M	840	⊕VA-VAE [46]	2	**1.42**	675M	432
⊕DC-AE [4]	2	**1.75**	675M	840	⊕VA-VAE [46]	1	2.19	675M	432
⊕SD-VAE [32]	1	2.63	675M	360	⊕SD-VAE [32]	1	2.10	675M	424
⊕SD-VAE [32]	2	2.11	675M	360	⊕E2E-VAE [23]	1	2.29	675M	264

(b) When applied to the flow-based models, such as a pre-trained DDT-XL/2 model, UCGM-S achieved an FID of 1.25 with 200 NFEs, compared to the original 1.28 FID requiring 500 NFEs. This demonstrates a performance improvement achieved alongside enhanced efficiency.

This approach generalizes across different generative model frameworks and resolutions. For instance, on 256×256 resolution using the flow-based REPA-E-XL model, UCGM-S attained 1.06 FID at 80 NFEs, which surpasses the baseline performance of 1.26 FID achieved at 500 NFEs.

In summary, UCGM-*S acts as a broadly applicable technique for efficient sampling, demonstrating cases where performance (FID) improves despite a reduction in sampling steps*.

UCGM-T + UCGM-S: Synergistic efficiency. The combination of UCGM-T training and UCGM-S sampling yields highly competitive generative performance with minimal NFEs:

(a) 512×512: With a DC-AE autoencoder, our framework achieved 1.48 FID at 40 NFEs. This outperforms DiT-XL/1⊕DC-AE (2.41 FID, 500 NFEs) and EDM2-XXL (1.91 FID, 63 NFEs), with comparable or reduced model size.

(b) 256×256: With an E2E-VAE autoencoder, we attained 1.21 FID at 40 NFEs. This result exceeds prior SOTA models like MAR-H (1.55 FID, 512 NFEs) and REPA-E-XL (1.26 FID, 500 NFEs).

Importantly, models trained with UCGM-T maintain robustness under extremely low-step sampling regimes. At 20 NFEs, the 256×256 performance degrades gracefully to 1.30 FID, a result that still exceeds the performance of several baseline models operating at significantly higher NFEs.

In summary, *the demonstrated robustness and efficiency of* UCGM-*{T, S} across various scenarios underscore the high potential of our* UCGM *for multi-step continuous generative modeling*.

4.3 Comparison with SOTA Methods for Few-Step Generation

As evidenced by the results in Tab. 4, our UCGM-{T, S} framework exhibits superior performance across two key settings: $\lambda = 0$, characteristic of a multi-step regime akin to diffusion and flow-matching models, and $\lambda = 1$, indicative of a few-step regime resembling consistency models.

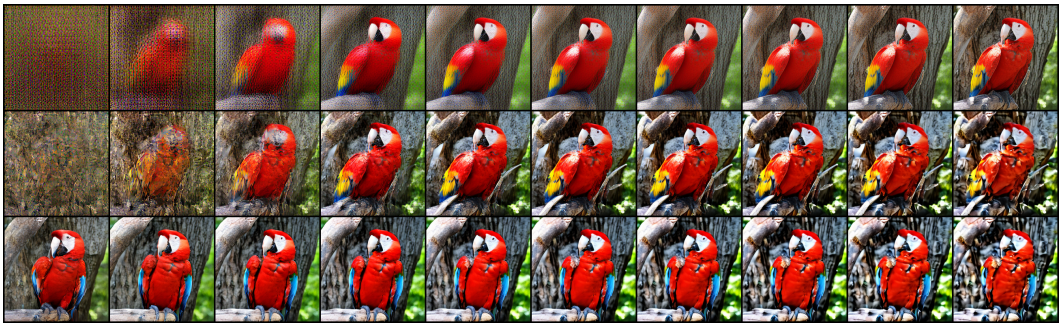
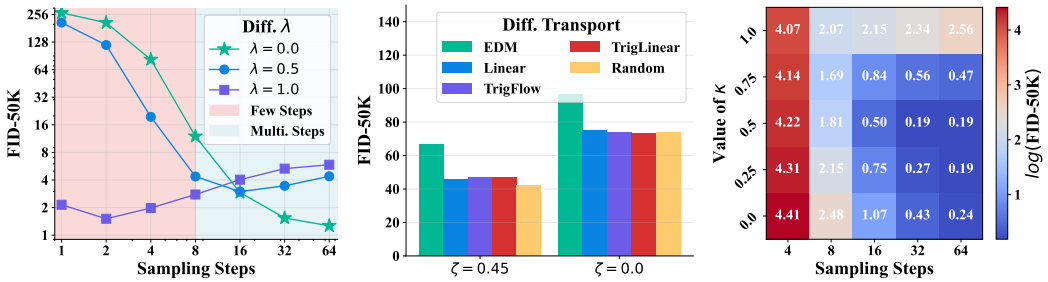


Figure 2: **Intermediate images generated during 60-step sampling from UCGM-S.** Columns display intermediate images $\hat{\mathbf{x}}_t$ produced at different timesteps t during a single sampling trajectory, ordered from left to right by decreasing t. Rows correspond to models trained with $\lambda \in \{0.0, 0.5, 1.0\}$, ordered from top to bottom.



(a) **Various λ and sampling steps.** (b) **Different ζ and transport types.** (c) **Various κ and sampling steps.**

Figure 3: **Ablation studies of UCGM on ImageNet-1K 256×256.** These studies evaluate key factors of the proposed UCGM. Ablations presented in (a) and (c) utilize XL/1 models with the VA-VAE autoencoder. For the results shown in (b), B/2 models with the SD-VAE autoencoder are used to facilitate more efficient training.

Few-step regime ($\lambda = 1$). Configured for few-step generation, UCGM-{T, S} achieves SOTA sample quality with minimal NFEs, surpassing existing specialized consistency models and GANs:

(a) 512×512: Using a DC-AE autoencoder, our model achieves an FID of 1.75 with 2 NFEs and 675M parameters. This outperforms sCD-XXL, a leading consistency distillation model, which reports 1.88 FID with 2 NFEs and 1.5B parameters.

(b) 256×256: Using a VA-VAE autoencoder, our model achieves an FID of 1.42 with 2 NFEs. This is a notable improvement over IMM-XL/2, which obtains 1.99 FID with $8 \times 2 = 16$ NFEs, demonstrating higher sample quality while requiring $8\times$ fewer sampling steps.

In summary, *these results demonstrate the capability of* UCGM-*{T, S} to deliver high-quality generation with minimal sampling cost, which is advantageous for practical applications.*

Multi-step regime ($\lambda = 0$). In the multi-step regime ($\lambda = 0$), where models is optimized for multi-step sampling, it nonetheless demonstrates competitive performance even when utilizing a moderate number of sampling steps.

(a) 512×512: Using a DC-AE autoencoder, our model obtains an FID of 1.81 with 16 NFEs and 675M parameters. This result is competitive with or superior to existing methods such as VAR-$d30$-s, which reports 2.63 FID with $10 \times 2 = 20$ NFEs and 2.3B parameters.

(b) 256×256: Using an E2E-VAE autoencoder, our model achieves an FID of 1.40 with 16 NFEs. This surpasses IMM-XL/2, which obtains 1.99 FID with $8 \times 2 = 16$ NFEs, demonstrating improved quality at the same sampling cost.

In summary, *our* UCGM-*{T, S} framework demonstrates versatility and high performance across both few-step ($\lambda = 1$) and multi-step ($\lambda = 0$) sampling regimes. As shown, it consistently achieves SOTA or competitive sample quality relative to existing methods, often requiring fewer sampling steps or parameters, which are important factors for efficient high-resolution image synthesis.*

4.4 Ablation Study over the Key Factors of UCGM

Unless otherwise specified, experiments in this section are conducted with $\kappa = 0.0$ and $\lambda = 0.0$.

Effect of λ in UCGM-T. Fig. 3a demonstrates that varying λ influences the range of effective sampling steps for trained models. For instance, with $\lambda = 1$, optimal performance is attained at 2 sampling steps. In contrast, with $\lambda = 0.5$, optimal performance is observed at 16 steps. To investigate this phenomenon, we visualize intermediate samples generated during the sampling process (Alg. 2). Fig. 2 demonstrates: (a) For $\lambda = 1.0$, high visual fidelity is achieved early in the sampling process. (b) In contrast, for $\lambda = 0.5$, high visual fidelity emerges in the mid to late stages of sampling.

Impact of transport type in UCGM. The results in Fig. 3b demonstrates that UCGM-{T, S} is applicable with various transport types, albeit with some performance variation. Investigating these performance differences constitutes future work. The results also illustrate that the enhanced training objective (achieved with $\zeta = 0.45$ compared to $\zeta = 0.0$, per Sec. 3) consistently improves performance across all tested transport types.

Setting different κ in UCGM-S. Experimental results, depicted in Fig. 3c, illustrate the impact of κ on the trade-off between sampling steps and generation quality: (a) High κ values (e.g., 1.0 and 0.75) prove beneficial for extreme few-step sampling scenarios (e.g., 4 steps). (b) Moreover, mid-range κ values (0.25 to 0.5) achieve superior performance with fewer steps compared to $\kappa = 0.0$.

References

[1] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22669–22679, 2023.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[3] Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Enze Xie, and Song Han. Sana-sprint: One-step diffusion with continuous-time consistency distillation. *arXiv preprint arXiv:2503.09641*, 2025.

[4] Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. *arXiv preprint arXiv:2410.10733*, 2024.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[6] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[7] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[8] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[9] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

[10] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 23164–23173, 2023.

[11] Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. *arXiv preprint arXiv:2406.14548*, 2024.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[15] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[16] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[17] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10124–10134, 2023.

[18] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[19] Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. *Advances in Neural Information Processing Systems*, 37:52996–53021, 2024.

[20] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24174–24184, 2024.

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[22] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html*, 6(1):1, 2009.

[23] Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. Repa-e: Unlocking vae for end-to-end tuning with latent diffusion transformers. *arXiv preprint arXiv:2504.10483*, 2025.

[24] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024.

[25] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[26] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[28] Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024.

[29] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pages 23–40. Springer, 2024.

[30] A Paszke. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[31] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[33] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022.

[34] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[35] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[36] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

[37] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[38] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[39] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[40] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

[41] Zhicong Tang, Jianmin Bao, Dong Chen, and Baining Guo. Diffusion models without classifier-free guidance. *arXiv preprint arXiv:2502.12154*, 2025.

[42] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[44] Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. *arXiv preprint arXiv:2504.05741*, 2025.

[45] Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint arXiv:2410.10629*, 2024.

[46] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. *arXiv preprint arXiv:2501.01423*, 2025.

[47] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023.

[48] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

[49] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

[50] Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. *arXiv preprint arXiv:2306.09305*, 2023.

[51] Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. *arXiv preprint arXiv:2503.07565*, 2025.

Contents

A Detailed Experiment

A.1 Detailed Experimental Setting

A.1.1 Detailed Datasets

Image datasets. We conduct experiments on three datasets: CIFAR-10 [21], ImageNet-1K [5]:

(a) CIFAR-10 is a widely used benchmark dataset for image classification and generation tasks. It consists of $60,000$ color images, each with a resolution of 32×32 pixels, categorized into 10 distinct classes. The dataset is divided into $50,000$ training images and $10,000$ test images.
(b) ImageNet-1K is a large-scale dataset containing over 1.2 million high-resolution images across $1,000$ categories.

Latent space datasets. However, directly training diffusion transformers in the pixel space is computationally expensive and inefficient. Therefore, following previous studies [48, 29], we train our diffusion transformers in latent space instead. Tab. 5 presents a comparative analysis of various Variational Autoencoder (VAE) architectures. SD-VAE is characterized by a higher spatial resolution in its latent representation (e.g., $\mathrm{H}/8 \times \mathrm{W}/8$) combined with a lower channel capacity (4 channels). Conversely, alternative models such as VA-VAE, E2E-VAE, and DC-AE achieve more significant spatial compression (e.g., $\mathrm{H}/16 \times \mathrm{W}/16$ or $\mathrm{H}/32 \times \mathrm{W}/32$) at the expense of an increased channel depth (typically 32 channels).

A key consideration is that the computational cost of a diffusion transformer subsequently processing these latent representations is primarily dictated by their spatial dimensions, rather than their channel capacity [4]. Specifically, if the latent map is processed by a transformer by dividing it into non-overlapping patches, the cost is proportional to the number of these patches. This quantity is given by $(\mathrm{H}/\text{Compression Ratio}/\text{Patch Size}) \times (\mathrm{W}/\text{Compression Ratio}/\text{Patch Size})$. Here, H and W are the input image dimensions, Compression Ratio refers to the spatial compression factor of the VAE (e.g., 8, 16, 32 as detailed in Tab. 5), and Patch Size denotes the side length of the patches processed by the transformer.

Table 5: **Comparison of different VAE architectures in terms of latent space dimensions and channel capacity.** The table contrasts four variational autoencoder variants (SD-VAE, VA-VAE, E2E-VAE, and DC-AE) by their spatial compression ratios (latent size) and feature channel dimensions. Here, H and W denote input image height and width (e.g., 256×256 or 512×512), respectively.

	SD-VAE (`ema`, `mse`) [32]	VA-VAE [46]	E2E-VAE [23]	DC-AE (*f32c32*) [4]
Latent Size	$(\mathrm{H}/8) \times (\mathrm{W}/8)$	$(\mathrm{H}/16) \times (\mathrm{W}/16)$	$(\mathrm{H}/16) \times (\mathrm{W}/16)$	$(\mathrm{H}/32) \times (\mathrm{W}/32)$
Channels	4	32	32	32

A.1.2 Detailed Neural Architecture

Diffusion Transformers (DiTs) represent a paradigm shift in generative modeling by replacing the traditional U-Net backbone with a Transformer-based architecture. Proposed by *Scalable Diffusion Models with Transformers* [31], DiTs exhibit superior scalability and performance in image generation tasks. In this paper, we utilize three key variants—DiT-B (130M parameters), DiT-L (458M parameters), and DiT-XL (675M parameters).

To improve training stability, informed by recent studies [46, 44], we incorporate several architectural modifications into the DiT framework: (a) SwiGLU feed-forward networks (FFN) [34]; (b) RMSNorm [49] without learnable affine parameters; (c) Rotary Positional Embeddings (RoPE) [39]; and (d) parameter-free RMSNorm applied to Key (K) and Query (Q) projections in self-attention layers [43]. This modified architecture, termed TiT, reflects these integrated enhancements.

B Theoretical Analysis

B.1 Main Results

B.1.1 Unifed Training Objective

The unifying loss $\mathcal{L}(\boldsymbol{\theta})$ is formulated as:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{z},\mathbf{x})\sim p(\mathbf{z},\mathbf{x}),t}\left[\frac{1}{\hat{\omega}(t)}\left\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{\lambda t},\lambda t),\mathbf{x}_{\lambda t},\lambda t)\right\|_2^2\right]$$

We are interested in the gradient of the term inside the expectation with respect to $\boldsymbol{\theta}$. Let this inner term be $J(\boldsymbol{\theta})$:

$$J(\boldsymbol{\theta}) = \frac{1}{\hat{\omega}(t)}\left\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{\lambda t},\lambda t),\mathbf{x}_{\lambda t},\lambda t)\right\|_2^2$$

Using the identity $\nabla_{\boldsymbol{\theta}}\|\mathbf{v}(\boldsymbol{\theta})\|_2^2 = 2\left(\nabla_{\boldsymbol{\theta}}\mathbf{v}(\boldsymbol{\theta})\right)^T\mathbf{v}(\boldsymbol{\theta})$, which we can write using inner product notation as $2\langle\nabla_{\boldsymbol{\theta}}\mathbf{v}(\boldsymbol{\theta}),\mathbf{v}(\boldsymbol{\theta})\rangle$. Here, $\nabla_{\boldsymbol{\theta}}\mathbf{v}(\boldsymbol{\theta})$ denotes the Jacobian of $\mathbf{v}$ with respect to $\boldsymbol{\theta}$, and the inner product notation $\langle J,\mathbf{w}\rangle$ is understood as $J^T\mathbf{w}$ when J is a Jacobian, yielding the gradient vector.

Let $\mathbf{A}(\boldsymbol{\theta}) = \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t)$ and $\mathbf{B}(\boldsymbol{\theta}^-) = \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{\lambda t},\lambda t),\mathbf{x}_{\lambda t},\lambda t)$. The gradient of the squared norm $\|\mathbf{A}(\boldsymbol{\theta}) - \mathbf{B}(\boldsymbol{\theta}^-)\|_2^2$ is:

$$\nabla_{\boldsymbol{\theta}}\|\mathbf{A}(\boldsymbol{\theta}) - \mathbf{B}(\boldsymbol{\theta}^-)\|_2^2 = 2\left\langle\nabla_{\boldsymbol{\theta}}\left[\mathbf{A}(\boldsymbol{\theta}) - \mathbf{B}(\boldsymbol{\theta}^-)\right],\mathbf{A}(\boldsymbol{\theta}) - \mathbf{B}(\boldsymbol{\theta}^-)\right\rangle$$

The term $\mathbf{B}(\boldsymbol{\theta}^-)$ depends on $\boldsymbol{\theta}^-$, which are fixed parameters from a previous iteration. Thus, $\mathbf{B}(\boldsymbol{\theta}^-)$ is treated as a constant with respect to the current parameters $\boldsymbol{\theta}$. This is equivalent to applying a stop-gradient operator to $\mathbf{B}(\boldsymbol{\theta}^-)$. So, $\nabla_{\boldsymbol{\theta}}\mathbf{B}(\boldsymbol{\theta}^-) = \mathbf{0}$. The gradient expression simplifies to:

$$\nabla_{\boldsymbol{\theta}}\|\mathbf{A}(\boldsymbol{\theta}) - \mathbf{B}(\boldsymbol{\theta}^-)\|_2^2 = 2\left\langle\nabla_{\boldsymbol{\theta}}\mathbf{A}(\boldsymbol{\theta}),\mathbf{A}(\boldsymbol{\theta}) - \mathbf{B}(\boldsymbol{\theta}^-)\right\rangle \tag{7}$$

This is $2\left\langle\nabla_{\boldsymbol{\theta}}\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t),\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{\lambda t},\lambda t),\mathbf{x}_{\lambda t},\lambda t)\right\rangle$.

We define $\Delta\boldsymbol{f}_t^{\mathbf{x}}$ as a finite difference approximation involving only terms dependent on the fixed parameters $\boldsymbol{\theta}^-$:

$$\Delta\boldsymbol{f}_t^{\mathbf{x}} := \frac{\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t),\mathbf{x}_t,t) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{\lambda t},\lambda t),\mathbf{x}_{\lambda t},\lambda t)}{t-\lambda t}, \quad \lambda\in(0,1)$$

Note that in this definition, $\mathbf{x}_t^\star$ and $\mathbf{x}_{\lambda t}^\star$ from the original prompt have been interpreted as $\mathbf{x}_t$ and $\mathbf{x}_{\lambda t}$ respectively, to align with the arguments in the loss function. Critically, $\Delta\boldsymbol{f}_t^{\mathbf{x}}$ does not depend on $\boldsymbol{\theta}$.

To relate (7) to $\Delta\boldsymbol{f}_t^{\mathbf{x}}$, an approximation is made: $\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t) \approx \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t),\mathbf{x}_t,t)$. This implies that the current model's output $\mathbf{A}(\boldsymbol{\theta})$ is close to the output using the target parameters $\boldsymbol{\theta}^-$ at time t. Under this approximation, the term $\mathbf{A}(\boldsymbol{\theta}) - \mathbf{B}(\boldsymbol{\theta}^-)$ in (7) becomes:

$$\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{\lambda t},\lambda t),\mathbf{x}_{\lambda t},\lambda t) \approx \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t),\mathbf{x}_t,t) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{\lambda t},\lambda t),\mathbf{x}_{\lambda t},\lambda t)$$

The right hand side is $(t-\lambda t)\Delta\boldsymbol{f}_t^{\mathbf{x}}$. So, (7) is approximated as:

$$2\left\langle\nabla_{\boldsymbol{\theta}}\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t),(t-\lambda t)\Delta\boldsymbol{f}_t^{\mathbf{x}}\right\rangle = 2(t-\lambda t)\left\langle\nabla_{\boldsymbol{\theta}}\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t),\Delta\boldsymbol{f}_t^{\mathbf{x}}\right\rangle$$

Since $\Delta\boldsymbol{f}_t^{\mathbf{x}}$ is independent of $\boldsymbol{\theta}$, we can write this as:

$$2(t-\lambda t)\nabla_{\boldsymbol{\theta}}\left\langle\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t),\Delta\boldsymbol{f}_t^{\mathbf{x}}\right\rangle$$

The factor $2(t-\lambda t)$ is a scalar that can be absorbed into the learning rate. Thus, the gradient of $J(\boldsymbol{\theta})$ is considered proportional to $\frac{1}{\hat{\omega}(t)}\nabla_{\boldsymbol{\theta}}\left\langle\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t),\Delta\boldsymbol{f}_t^{\mathbf{x}}\right\rangle$.

Now, we substitute the specific form of $\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_t,\mathbf{x}_t,t)$:

$$\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t) = \frac{\alpha(t)\cdot\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - \hat{\alpha}(t)\cdot\mathbf{x}_t}{\alpha(t)\cdot\hat{\gamma}(t) - \hat{\alpha}(t)\cdot\gamma(t)}$$

Let $D(t) = \alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t)$, which is a scalar independent of $\boldsymbol{\theta}$. The gradient term becomes proportional to:

$$\frac{1}{\hat{\omega}(t)} \nabla_{\boldsymbol{\theta}} \left\langle \frac{\alpha(t) \cdot \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \hat{\alpha}(t) \cdot \mathbf{x}_t}{D(t)}, \Delta \boldsymbol{f}_t^{\mathbf{x}} \right\rangle$$

Given $\hat{\omega}(t) = \frac{\tan(t)}{4}$, so $\frac{1}{\hat{\omega}(t)} = \frac{4}{\tan(t)}$. The expression is:

$$\frac{4}{\tan(t)} \nabla_{\boldsymbol{\theta}} \left\langle \frac{\alpha(t) \cdot \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \hat{\alpha}(t) \cdot \mathbf{x}_t}{D(t)}, \Delta \boldsymbol{f}_t^{\mathbf{x}} \right\rangle$$

We can take constants out of the gradient operator and the inner product:

$$\begin{aligned}
&\frac{4}{\tan(t) D(t)} \nabla_{\boldsymbol{\theta}} \left\langle \alpha(t) \cdot \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \hat{\alpha}(t) \cdot \mathbf{x}_t, \Delta \boldsymbol{f}_t^{\mathbf{x}} \right\rangle \\
&= \frac{4}{\tan(t) D(t)} \nabla_{\boldsymbol{\theta}} \left(\left\langle \alpha(t) \cdot \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \Delta \boldsymbol{f}_t^{\mathbf{x}} \right\rangle - \left\langle \hat{\alpha}(t) \cdot \mathbf{x}_t, \Delta \boldsymbol{f}_t^{\mathbf{x}} \right\rangle \right)
\end{aligned}$$

Since $\hat{\alpha}(t) \cdot \mathbf{x}_t$ and $\Delta \boldsymbol{f}_t^{\mathbf{x}}$ are independent of $\boldsymbol{\theta}$, the term $\langle \hat{\alpha}(t) \cdot \mathbf{x}_t, \Delta \boldsymbol{f}_t^{\mathbf{x}} \rangle$ is constant with respect to $\boldsymbol{\theta}$, so its gradient is zero. This is the justification for "constants like $\mathbf{x}_t$ can be dropped" when they appear as additive terms inside the gradient of an inner product. The expression simplifies to:

$$\frac{4}{\tan(t) D(t)} \nabla_{\boldsymbol{\theta}} \left\langle \alpha(t) \cdot \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \Delta \boldsymbol{f}_t^{\mathbf{x}} \right\rangle = \frac{4\alpha(t)}{\tan(t) D(t)} \nabla_{\boldsymbol{\theta}} \left\langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \Delta \boldsymbol{f}_t^{\mathbf{x}} \right\rangle$$

Substituting $\tan(t) = \sin(t)/\cos(t)$ and $D(t) = \alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t)$:

$$\frac{4\alpha(t)\cos(t)}{\sin(t)(\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t))} \nabla_{\boldsymbol{\theta}} \left\langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \Delta \boldsymbol{f}_t^{\mathbf{x}} \right\rangle$$

Since $\Delta \boldsymbol{f}_t^{\mathbf{x}}$ is independent of $\boldsymbol{\theta}$, $\nabla_{\boldsymbol{\theta}} \langle \boldsymbol{F}_{\boldsymbol{\theta}}, \Delta \boldsymbol{f}_t^{\mathbf{x}} \rangle = \langle \nabla_{\boldsymbol{\theta}} \boldsymbol{F}_{\boldsymbol{\theta}}, \Delta \boldsymbol{f}_t^{\mathbf{x}} \rangle$. The expression is:

$$\left\langle \nabla_{\boldsymbol{\theta}} \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \frac{4\alpha(t)\cos(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t) \cdot (\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t))} \right\rangle$$

This can be rewritten by moving $\cos(t)$ and other $\boldsymbol{\theta}$-independent scalars outside the $\nabla_{\boldsymbol{\theta}}$ operator if it's applied to the entire inner product, or inside the second argument of the inner product if $\nabla_{\boldsymbol{\theta}}$ applies only to $\boldsymbol{F}_{\boldsymbol{\theta}}$. The target expression is:

$$\nabla_{\boldsymbol{\theta}} \left(\cos(t) \left\langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t) \cdot (\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t))} \right\rangle \right)$$

This is justified because $\cos(t)$ and the term $\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t) \cdot (\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t))}$ (let's call it $\mathbf{K}_t$) are independent of $\boldsymbol{\theta}$. So,

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \left(\cos(t) \langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \mathbf{K}_t \rangle \right) &= \cos(t) \nabla_{\boldsymbol{\theta}} \langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \mathbf{K}_t \rangle \\
&= \cos(t) \langle \nabla_{\boldsymbol{\theta}} \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \mathbf{K}_t \rangle \\
&= \langle \nabla_{\boldsymbol{\theta}} \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \cos(t) \mathbf{K}_t \rangle \\
&= \left\langle \nabla_{\boldsymbol{\theta}} \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \cos(t) \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t) \cdot (\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t))} \right\rangle
\end{aligned}$$

This matches the derived form. We can also view the training objective as:

$$\mathbb{E}_{(\mathbf{z},\mathbf{x}) \sim p(\mathbf{z},\mathbf{x}), t} \left[\cos(t) \| \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t) + \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t) \cdot (\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t))} \|_2^2 \right] ,$$

which would induce the same gradient, owing to the analysis below:

The first expression is:

$$\nabla_{\boldsymbol{\theta}} \cos(t) \left\langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t) \cdot (\alpha(t) \cdot \hat{\gamma}(t) - \hat{\alpha}(t) \cdot \gamma(t))} \right\rangle$$

Since $\cos(t)$ and the term $\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}$ do not depend on $\boldsymbol{\theta}$, we can write this as:

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}}\cos(t)\left\langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\rangle \\
&= \cos(t)\nabla_{\boldsymbol{\theta}}\left\langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\rangle \\
&= \cos(t)\left(\nabla_{\boldsymbol{\theta}}\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\right)^T\left(\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right) \quad (*)
\end{aligned}$$

The second expression is the training objective:

$$\mathbb{E}_{(\mathbf{z},\mathbf{x})\sim p(\mathbf{z},\mathbf{x}),t}\left[\cos(t)\left\|\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)+\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\|_2^2\right]$$

Let us consider the term inside the expectation:

$$\cos(t)\left\|\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)+\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\|_2^2$$

Expanding the squared L2-norm, $\|\mathbf{a}+\mathbf{b}\|_2^2 = \|\mathbf{a}\|_2^2 + \|\mathbf{b}\|_2^2 + 2\langle \mathbf{a},\mathbf{b}\rangle$:

$$\begin{aligned}
&\cos(t)\left\|\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)+\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\|_2^2 \\
&= \cos(t)\Bigg[\left\|\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)\right\|_2^2 \\
&\qquad + \left\|\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\|_2^2 \\
&\qquad + 2\left\langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\rangle\Bigg] \\
&= \cos(t)\left\|\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)\right\|_2^2 \\
&\quad + \cos(t)\left\|\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\|_2^2 \\
&\quad + 2\cos(t)\left\langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\rangle \\
&\quad - 2\cos(t)\left\langle \boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\rangle
\end{aligned}$$

Now, we compute the gradient of this entire expression with respect to $\boldsymbol{\theta}$:

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}}\Bigg[\cos(t)\left\|\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)\right\|_2^2 \\
&\qquad + \cos(t)\left\|\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\|_2^2 \\
&\qquad + 2\cos(t)\left\langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\rangle \\
&\qquad - 2\cos(t)\left\langle \boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\rangle\Bigg]
\end{aligned}$$

Applying the gradient operator term by term:

(a) The gradient of $\cos(t)\left\|\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)\right\|_2^2$ is
$2\cos(t)\left(\nabla_{\boldsymbol{\theta}}\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\right)^T\left(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)\right)$, since $\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)$ is constant w.r.t. $\boldsymbol{\theta}$.

(b) The gradient of $\cos(t)\left\|\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\|_2^2$ is $\mathbf{0}$, as this term is constant w.r.t. $\boldsymbol{\theta}$.

(c) The gradient of $2\cos(t)\left\langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\rangle$ is

$$2\cos(t)\nabla_{\boldsymbol{\theta}}\left\langle \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\rangle$$
$$= 2\cos(t)\left(\nabla_{\boldsymbol{\theta}}\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\right)^T\left(\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right).$$

(d) The gradient of $-2\cos(t)\left\langle \boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t), \frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right\rangle$ is $\mathbf{0}$, as this term is constant w.r.t. $\boldsymbol{\theta}$.

Summing these results, the gradient of the term inside the expectation is:

$$2\cos(t)\left(\nabla_{\boldsymbol{\theta}}\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\right)^T\left(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)\right)$$
$$+\,2\cos(t)\left(\nabla_{\boldsymbol{\theta}}\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\right)^T\left(\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right)$$

Comparing the second term of this sum with the result $(*)$ for the first expression:

$$2\cos(t)\left(\nabla_{\boldsymbol{\theta}}\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\right)^T\left(\frac{4\alpha(t)\Delta \boldsymbol{f}_t^{\mathbf{x}}}{\sin(t)\cdot(\alpha(t)\cdot\hat{\gamma}(t)-\hat{\alpha}(t)\cdot\gamma(t))}\right)$$

is exactly $2\times$ (first expression evaluated at $(*)$). Thus, the gradient of the integrand of the training objective contains the first expression multiplied by a factor of 2.

B.1.2 Learning Objective as $\lambda \to 1$

Since

$$\begin{aligned}\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_0,\mathbf{x}_0,0) &= \frac{\alpha(0)\cdot\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_0,0)-\hat{\alpha}(0)\cdot\mathbf{x}_0}{\alpha(0)\cdot\hat{\gamma}(0)-\hat{\alpha}(0)\cdot\gamma(0)}\\ &= \frac{0\cdot\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_0,0)-\hat{\alpha}(0)\cdot\mathbf{x}_0}{0\cdot\hat{\gamma}(0)-\hat{\alpha}(0)\cdot 1}\\ &= \frac{\mathbf{0}-\hat{\alpha}(0)\cdot\mathbf{x}_0}{0-\hat{\alpha}(0)}\\ &= \mathbf{x}_0\end{aligned}$$

$\boldsymbol{f}^{\mathbf{x}}$ satisfies the boundary condition of consistency model (CM). Also note that our unified training objective is formally similar to that of consistency model, except that we use λt instead of $t-\Delta t$. When $\lambda \to 1$(i.e. $\Delta t \to 0$), we get so-called continous consistency model[37, 28], and the term $\Delta \boldsymbol{f}_t^{\mathbf{x}}$ in $\boldsymbol{F}_t^{\text{target}}$ tends to $\frac{\mathrm{d}\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t),\mathbf{x}_t,t)}{\mathrm{d}t}$. In this case, we could use some numerical methods to estimate this derivative, for instance through the symmetric difference quotient:

$$\frac{\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t+\epsilon},t+\epsilon),\mathbf{x}^{\star}_{t+\epsilon},t+\epsilon)-\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t-\epsilon},t-\epsilon),\mathbf{x}^{\star}_{t-\epsilon},t-\epsilon)}{2\epsilon}.$$

Remark. To better understand the unified loss, let's analyze a case when coefficients $\hat{\alpha},\hat{\gamma}$ satisfy the condition of flow model. For simplicity we use the notation $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) := \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t)$, the training objective is then equal to

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{t,(\mathbf{z},\mathbf{x})}\left[\frac{1}{\hat{\omega}(t)}\|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{\lambda t},\lambda t)\|_2^2\right].$$

where $(\mathbf{z},\mathbf{x})\sim p(\mathbf{z},\mathbf{x}),\mathbf{x}_t|(\mathbf{z},\mathbf{x}) = \alpha(t)\mathbf{z}+\gamma(t)\mathbf{x}$.

Let $\phi_t(\mathbf{x})$ be the solution of the flow ODE determined by the velocity field $\mathbf{v}^*(\mathbf{x}_t,t) = \mathbb{E}_{(\mathbf{z},\mathbf{x})|\mathbf{x}_t}\left[\alpha'(t)\mathbf{z}+\gamma'(t)\mathbf{x}|\mathbf{x}_t\right]$ and an initial value $\mathbf{x}$. For given $\mathbf{x}_t$, let $\mathbf{x}^* = \phi_t^{-1}(\mathbf{x}_t)$ and $\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t) = \boldsymbol{f}_{\boldsymbol{\theta}}(\phi_t(\mathbf{x}^*),t)$. When $\lambda\to 1$, the gradient of the loss tends to

$$\begin{aligned}\lim_{\lambda\to 1}\nabla_{\boldsymbol{\theta}}\frac{\mathcal{L}(\boldsymbol{\theta})}{2(1-\lambda)} &= \mathbb{E}_t\left[\frac{t}{\hat{\omega}(t)}\cdot\mathbb{E}_{(\mathbf{z},\mathbf{x})}\lim_{\lambda\to 1}\left\langle\frac{\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{\lambda t},\lambda t)}{t-\lambda t},\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\right\rangle\right]\\ &= \mathbb{E}_t\left[\frac{t}{\hat{\omega}(t)}\cdot\mathbb{E}_{(\mathbf{z},\mathbf{x})}\left\langle\frac{\mathrm{d}\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\mathrm{d}t},\nabla_{\boldsymbol{\theta}}\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t)\right\rangle\right]\end{aligned}$$

The inner expectation can be computed as:

$$\begin{aligned}
\mathbb{E}_{(\mathbf{z},\mathbf{x}),\mathbf{x}_t}\ \langle \frac{d\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{dt}, \nabla_{\boldsymbol{\theta}}\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t)\rangle &= \mathbb{E}_{(\mathbf{z},\mathbf{x}),\mathbf{x}_t}\ \langle \partial_1 \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\cdot\frac{d\mathbf{x}_t}{dt} + \partial_2 \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t), \nabla_{\boldsymbol{\theta}}\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t)\rangle \\
&= \mathbb{E}_{(\mathbf{z},\mathbf{x}),\mathbf{x}_t}\ \langle \partial_1 \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\cdot(\alpha'(t)\mathbf{z}+\gamma'(t)\mathbf{x}) + \partial_2 \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t), \nabla_{\boldsymbol{\theta}}\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t)\rangle \\
&= \mathbb{E}_{\mathbf{x}_t}\left[\mathbb{E}_{(\mathbf{z},\mathbf{x})|\mathbf{x}_t}\ \langle \partial_1 \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\cdot(\alpha'(t)\mathbf{z}+\gamma'(t)\mathbf{x}) + \partial_2 \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t), \nabla_{\boldsymbol{\theta}}\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t)\rangle\right] \\
&= \mathbb{E}_{\mathbf{x}_t}\ \langle \partial_1 \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\cdot\mathbb{E}_{(\mathbf{z},\mathbf{x})|\mathbf{x}_t}\left[\alpha'(t)\mathbf{z}+\gamma'(t)\mathbf{x}|\mathbf{x}_t\right] + \partial_2 \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t), \nabla_{\boldsymbol{\theta}}\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t)\rangle \\
&= \mathbb{E}_{\mathbf{x}_t}\ \langle \partial_1 \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\cdot\mathbf{v}^*(\mathbf{x}_t,t) + \partial_2 \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t), \nabla_{\boldsymbol{\theta}}\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t)\rangle \\
&= \mathbb{E}_{\mathbf{x}_t}\ \langle \frac{\mathrm{d}\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t)}{\mathrm{d}t}, \nabla_{\boldsymbol{\theta}}\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t)\rangle \\
&= \nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathbf{x}^*}\ \frac{1}{2}\|\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t) - \boldsymbol{g}_{\boldsymbol{\theta}^-}(\mathbf{x}^*,t) + \frac{\mathrm{d}\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t)}{\mathrm{d}t}\|_2^2
\end{aligned}$$

Thus from the perspective of gradient, when $\lambda \to 1$ the training objective is equivalent to

$$\mathbb{E}_{\mathbf{x}^*,t}\left[\frac{t}{\hat{\omega}(t)}\cdot\|\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t) - \boldsymbol{g}_{\boldsymbol{\theta}^-}(\mathbf{x}^*,t) + \frac{\mathrm{d}\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t)}{\mathrm{d}t}\|_2^2\right]$$

which naturally leads to the solution $\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,t) = \mathbf{x}^*$ (since $\boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{x}^*,0) \equiv \mathbf{x}^*$), or equivalently $\boldsymbol{f}_{\boldsymbol{\theta}^*}(\mathbf{x}_t,t) = \phi_t^{-1}(\mathbf{x}_t)$, that is the definition of consistency function.

B.1.3 Learning Objective as $\lambda \to 0$

When $\lambda = 0$, the training objective is just

$$\mathbb{E}_{(\mathbf{z},\mathbf{x})\sim p(\mathbf{z},\mathbf{x}),t}\left[\frac{1}{\hat{\omega}(t)}\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t),\mathbf{x}_t,t) - \mathbf{x}\|_2^2\right]$$

which is the standard x-prediction loss for a regular diffusion model.

Let's further expand the inner expression:

$$\begin{aligned}
\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_t,\mathbf{x}_t,t) - \mathbf{x} &= \frac{\alpha(t)\cdot\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - \hat{\alpha}(t)\cdot\mathbf{x}_t}{\alpha(t)\cdot\hat{\gamma}(t) - \hat{\alpha}(t)\cdot\gamma(t)} - \mathbf{x} \\
&= \frac{\alpha(t)\cdot\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - \hat{\alpha}(t)(\alpha(t)\cdot\mathbf{z}+\gamma(t)\cdot\mathbf{x}) - (\alpha(t)\cdot\hat{\gamma}(t) - \hat{\alpha}(t)\cdot\gamma(t))\cdot\mathbf{x}}{\alpha(t)\cdot\hat{\gamma}(t) - \hat{\alpha}(t)\cdot\gamma(t)} \\
&= \frac{\alpha(t)\cdot\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - \hat{\alpha}(t)\alpha(t)\cdot\mathbf{z} - \alpha(t)\hat{\gamma}(t)\cdot\mathbf{x}}{\alpha(t)\cdot\hat{\gamma}(t) - \hat{\alpha}(t)\cdot\gamma(t)} \\
&= \frac{\alpha(t)}{\alpha(t)\cdot\hat{\gamma}(t) - \hat{\alpha}(t)\cdot\gamma(t)}\left[\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - (\hat{\alpha}(t)\cdot\mathbf{z} + \cdot\hat{\gamma}(t)\cdot\mathbf{x})\right]
\end{aligned}$$

that's why in training algorithm when $\lambda = 0$ we can directly set

$$\boldsymbol{F}_t^{\text{target}} := \hat{\alpha}(t)\cdot\mathbf{z} + \cdot\hat{\gamma}(t)\cdot\mathbf{x}\,.$$

In this view it could be clearly seen that we are actually training a flow matching model when $\hat{\alpha}(t) = \frac{\mathrm{d}\alpha(t)}{\mathrm{d}t}$ and $\hat{\gamma}(t) = \frac{\mathrm{d}\gamma(t)}{\mathrm{d}t}$ are satisfied.

Remark. When $\lambda = 0$, we aim to derive the Probability Flow Ordinary Differential Equation (PF-ODE) corresponding to a defined forward process.

The forward process for a state variable $\mathbf{x}_t$ at time $t \in [0,1]$ is given by: $\mathbf{x}_t = \alpha(t)\mathbf{z} + \gamma(t)\mathbf{x}_0$, where $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ is a standard Gaussian random variable, and $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}_0)$ is drawn from the data distribution. The scalar functions $\alpha(t)$ and $\gamma(t)$ satisfy the boundary conditions: $\alpha(0) = \gamma(1) = 0$ and $\alpha(1) = \gamma(0) = 1$. This process can be associated with a Stochastic Differential Equation (SDE) of the form:

$$\mathrm{d}\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t,t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}_t,$$

where $\mathbf{w}_t$ is a standard Wiener process. We need to determine the vector-valued drift term $\mathbf{f}(\mathbf{x}_t,t)$ and the scalar diffusion term $g(t)$.

First, consider the conditional mean of the process $\mathbb{E}[\mathbf{x}_t|\mathbf{x}_0] = \gamma(t)\mathbf{x}_0$. Its time derivative is:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[\mathbf{x}_t|\mathbf{x}_0] = \gamma'(t)\mathbf{x}_0,$$

where $\gamma'(t) = \frac{\mathrm{d}\gamma(t)}{\mathrm{d}t}$. Assuming the drift of the SDE is linear in $\mathbf{x}_t$, i.e., $\mathbf{f}(\mathbf{x}_t, t) = H(t)\mathbf{x}_t$ for some matrix-valued function $H(t)$, the evolution of the SDE's conditional mean is $\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[\mathbf{x}_t|\mathbf{x}_0] = H(t)\mathbb{E}[\mathbf{x}_t|\mathbf{x}_0] = H(t)\gamma(t)\mathbf{x}_0$. Comparing this with $\gamma'(t)\mathbf{x}_0$, we find $H(t) = \frac{\gamma'(t)}{\gamma(t)}\mathbf{I}$ (for $\gamma(t) \neq 0$), where $\mathbf{I}$ is the identity matrix. Thus, the drift term is:

$$\mathbf{f}(\mathbf{x}_t, t) = \frac{\gamma'(t)}{\gamma(t)}\mathbf{x}_t.$$

Next, consider the conditional variance, $\mathrm{Var}(\mathbf{x}_t|\mathbf{x}_0) = \boldsymbol{\Sigma}(t) = \alpha(t)^2\mathbf{I}$. The evolution of this variance for the linear SDE is governed by the Lyapunov equation:

$$\frac{\mathrm{d}\boldsymbol{\Sigma}(t)}{\mathrm{d}t} = H(t)\boldsymbol{\Sigma}(t) + \boldsymbol{\Sigma}(t)H(t)^T + g(t)^2\mathbf{I}.$$

Substituting $\boldsymbol{\Sigma}(t) = \alpha(t)^2\mathbf{I}$ and $H(t) = \frac{\gamma'(t)}{\gamma(t)}\mathbf{I}$:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\alpha(t)^2\mathbf{I}) = \left(\frac{\gamma'(t)}{\gamma(t)}\mathbf{I}\right)(\alpha(t)^2\mathbf{I}) + (\alpha(t)^2\mathbf{I})\left(\frac{\gamma'(t)}{\gamma(t)}\mathbf{I}\right)^T + g(t)^2\mathbf{I}.$$

Since $\frac{\mathrm{d}}{\mathrm{d}t}(\alpha(t)^2) = 2\alpha(t)\alpha'(t)$ (where $\alpha'(t) = \frac{\mathrm{d}\alpha(t)}{\mathrm{d}t}$), the equation becomes:

$$2\alpha(t)\alpha'(t)\mathbf{I} = 2\frac{\gamma'(t)}{\gamma(t)}\alpha(t)^2\mathbf{I} + g(t)^2\mathbf{I}.$$

Solving for $g(t)^2$, we get:

$$g(t)^2 = 2\alpha(t)\alpha'(t) - 2\frac{\gamma'(t)}{\gamma(t)}\alpha(t)^2.$$

The Probability Flow ODE is given by the general form:

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}_t, t) - \frac{1}{2}g(t)^2\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t),$$

where $p_t(\mathbf{x}_t)$ is the marginal probability density of $\mathbf{x}_t$ at time t, and $\nabla_{\mathbf{x}_t}$ denotes the gradient with respect to $\mathbf{x}_t$. Substituting the derived expressions for $\mathbf{f}(\mathbf{x}_t, t)$ and $g(t)^2$:

$$\begin{aligned}\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} &= \left(\frac{\gamma'(t)}{\gamma(t)}\mathbf{x}_t\right) - \frac{1}{2}\left(2\alpha(t)\alpha'(t) - 2\frac{\gamma'(t)}{\gamma(t)}\alpha(t)^2\right)\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t) \\ &= \frac{\gamma'(t)}{\gamma(t)}\mathbf{x}_t - \left[\alpha(t)\alpha'(t) - \frac{\gamma'(t)}{\gamma(t)}\alpha^2(t)\right]\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t).\end{aligned}$$

Thus, the final PF-ODE is:

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \frac{\gamma'(t)}{\gamma(t)}\mathbf{x}_t - \left[\alpha(t)\alpha'(t) - \frac{\gamma'(t)}{\gamma(t)}\alpha^2(t)\right]\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)\,.$$

We show in below analysis that the above equation can be also be written as

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \alpha'(t)\cdot\boldsymbol{f}^{\mathbf{z}}_\star(\boldsymbol{F}_t, \mathbf{x}_t, t) + \gamma'(t)\cdot\boldsymbol{f}^{\mathbf{x}}_\star(\boldsymbol{F}_t, \mathbf{x}_t, t)\,.$$

where $\boldsymbol{f}^{\mathbf{x}}_\star, \boldsymbol{f}^{\mathbf{z}}_\star$ represent the analytical soulution for $\boldsymbol{f}^{\mathbf{x}}$ and $\boldsymbol{f}^{\mathbf{z}}$:

$$\boldsymbol{f}^{\mathbf{x}}_\star(\boldsymbol{F}_t, \mathbf{x}_t, t) = \mathbb{E}\left[\mathbf{x}|\mathbf{x}_t\right]\ ,\ \boldsymbol{f}^{\mathbf{z}}_\star(\boldsymbol{F}_t, \mathbf{x}_t, t) = \mathbb{E}\left[\mathbf{z}|\mathbf{x}_t\right]\ .$$

Specifically, Tweedie's formula states:

$$\boldsymbol{f}^{\mathbf{x}}_\star(\boldsymbol{F}_t, \mathbf{x}_t, t) = \frac{\mathbf{x}_t + \alpha^2(t)\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)}{\gamma(t)}\,.$$

Since the identity holds

$$\alpha(t)\cdot \boldsymbol{f}_\star^{\mathbf{z}}(\boldsymbol{F}_t,\mathbf{x}_t,t)+\gamma(t)\cdot \boldsymbol{f}_\star^{\mathbf{x}}(\boldsymbol{F}_t,\mathbf{x}_t,t)=\mathbf{x}_t\,,$$

for $\boldsymbol{f}_\star^{\mathbf{z}}$ we have:

$$\boldsymbol{f}_\star^{\mathbf{z}}(\boldsymbol{F}_t,\mathbf{x}_t,t)=-\alpha(t)\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)\,.$$

Now we can calculate the expression

$$\begin{aligned}\alpha'(t)\boldsymbol{f}_\star^{\mathbf{z}}(\boldsymbol{F}_t,\mathbf{x}_t,t)+\gamma'(t)\boldsymbol{f}_\star^{\mathbf{x}}(\boldsymbol{F}_t,\mathbf{x}_t,t)&=-\alpha'(t)\alpha(t)\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)+\gamma'(t)\frac{\mathbf{x}_t+\alpha^2(t)\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)}{\gamma(t)}\\&=\frac{\gamma'(t)}{\gamma(t)}\mathbf{x}_t-\left[\alpha(t)\alpha'(t)-\frac{\gamma'(t)}{\gamma(t)}\alpha^2(t)\right]\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)\,,\end{aligned}$$

which exactly matches the equation of PF-ODE as above.

Actually, this expression is also the velocity field of the flow ODE:

$$\begin{aligned}\mathbf{v}^*(\mathbf{x}_t,t)&=\mathbb{E}\left[\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t}|\mathbf{x}_t\right]\\&=\mathbb{E}\left[\alpha'(t)\mathbf{z}+\gamma'(t)\mathbf{x}|\mathbf{x}_t\right]\\&=\alpha'(t)\cdot\mathbb{E}\left[\mathbf{z}|\mathbf{x}_t\right]+\gamma'(t)\cdot\mathbb{E}\left[\mathbf{x}|\mathbf{x}_t\right]\\&=\alpha'(t)\cdot\boldsymbol{f}_\star^{\mathbf{z}}(\boldsymbol{F}_t,\mathbf{x}_t,t)+\gamma'(t)\cdot\boldsymbol{f}_\star^{\mathbf{x}}(\boldsymbol{F}_t,\mathbf{x}_t,t)\,.\end{aligned}$$

B.1.4 Enhanced Target Score

Recall that CFG proposes to modify the sampling distribution as

$$\tilde{p}_\theta(\mathbf{x}_t|\mathbf{c})\propto p_\theta(\mathbf{x}_t|\mathbf{c})p_\theta(\mathbf{c}|\mathbf{x}_t)^\zeta\,,$$

Bayesian rule gives

$$p_\theta(\mathbf{c}|\mathbf{x}_t)=\frac{p_\theta(\mathbf{x}_t|\mathbf{c})p_\theta(\mathbf{c})}{p_\theta(\mathbf{x}_t)}\,,$$

so we can futher deduce

$$\begin{aligned}\tilde{p}_\theta(\mathbf{x}_t|\mathbf{c})&\propto p_\theta(\mathbf{x}_t|\mathbf{c})p_\theta(\mathbf{c}|\mathbf{x}_t)^\zeta\\&=p_\theta(\mathbf{x}_t|\mathbf{c})(\frac{p_\theta(\mathbf{x}_t|\mathbf{c})p_\theta(\mathbf{c})}{p_\theta(\mathbf{x}_t)})^\zeta\\&\propto p_\theta(\mathbf{x}_t|\mathbf{c})(\frac{p_\theta(\mathbf{x}_t|\mathbf{c})}{p_\theta(\mathbf{x}_t)})^\zeta\,.\end{aligned}$$

When $t\in[0,s]$ ($s=0.75$), inspired by above expression and a recent work [41], we choose to use below as the target score function for training

$$\nabla_{\mathbf{x}_t}\log\left(p_t(\mathbf{x}_t|\mathbf{c})\left(\frac{p_{t,\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{c})}{p_{t,\boldsymbol{\theta}}(\mathbf{x}_t)}\right)^\zeta\right)$$

which equals to

$$\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t|\mathbf{c})+\zeta\left(\nabla_{\mathbf{x}_t}\log p_{t,\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{c})-\nabla_{\mathbf{x}_t}\log p_{t,\boldsymbol{\theta}}(\mathbf{x}_t)\right)\,.$$

For $\boldsymbol{f}_\star^{\mathbf{z}}$ we originally want to learn:

$$\boldsymbol{f}_\star^{\mathbf{z}}(\boldsymbol{F}_t,\mathbf{x}_t,t)=-\alpha(t)\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)\,,$$

now it turns to

$$\begin{aligned}\boldsymbol{f}_\star^{\mathbf{z}}(\boldsymbol{F}_t,\mathbf{x}_t,t)&=-\alpha(t)\nabla_{\mathbf{x}_t}\log\left(p_t(\mathbf{x}_t|\mathbf{c})\left(\frac{p_{t,\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{c})}{p_{t,\boldsymbol{\theta}}(\mathbf{x}_t)}\right)^\zeta\right)\\&=-\alpha(t)\left[\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t|\mathbf{c})+\zeta\left(\nabla_{\mathbf{x}_t}\log p_{t,\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{c})-\nabla_{\mathbf{x}_t}\log p_{t,\boldsymbol{\theta}}(\mathbf{x}_t)\right)\right]\\&=-\alpha(t)\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t|\mathbf{c})+\zeta\left(-\alpha(t)\nabla_{\mathbf{x}_t}\log p_{t,\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{c})+\alpha(t)\nabla_{\mathbf{x}_t}\log p_{t,\boldsymbol{\theta}}(\mathbf{x}_t)\right)\\&=-\alpha(t)\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t|\mathbf{c})+\zeta\left(\boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}_t,\mathbf{x}_t,t)-\boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}_t^{\varnothing},\mathbf{x}_t,t)\right)\,,\end{aligned}$$

thus in training we set the objective for $\boldsymbol{f}^{\mathbf{z}}$ as:

$$\mathbf{z}^\star \leftarrow \mathbf{z} + \zeta \cdot \left(\boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}_t, \mathbf{x}_t, t) - \boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}_t^{\varnothing}, \mathbf{x}_t, t)\right) .$$

Similarly, since $\boldsymbol{f}_\star^{\mathbf{x}} = \frac{\mathbf{x}_t + \alpha^2(t)\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)}{\gamma(t)}$ is also linear in the score function, we can use the same strategy to modify the training objective for $\boldsymbol{f}^{\mathbf{x}}$:

$$\mathbf{x}^\star \leftarrow \mathbf{x} + \zeta \cdot \left(\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_t, \mathbf{x}_t, t) - \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_t^{\varnothing}, \mathbf{x}_t, t)\right) .$$

When $t \in (s, 1]$ ($s = 0.75$), we further slightly modify the target score function to

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{c}) + \zeta \left(\nabla_{\mathbf{x}_t} \log p_{t,\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{c}) - \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right), \zeta = 0.5$$

which corresponds to the following training objective:

$$\mathbf{x}^\star \leftarrow \mathbf{x} + \frac{1}{2}\left(\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_t, \mathbf{x}_t, t) - \mathbf{x}\right), \mathbf{z}^\star \leftarrow \mathbf{z} + \frac{1}{2}\left(\boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F}_t, \mathbf{x}_t, t) - \mathbf{z}\right) .$$

B.1.5 Analysis on the Optimal Solution

Below we provide some examples to illustrate the property of the optimal solution for the unified loss by considering some simple cases of data distribution.

(for simplicity define $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \mathbf{x}_t, t)$)

Assume $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I})$ and consider a series of t together: $t = t_T > t_{T-1} > \ldots > t_1 > t_0 \approx 0$. This series could be obtained by $t_{j-1} = \lambda \cdot t_j, j = T, \ldots, 0$, for instance. With an abuse of notation, denote $\mathbf{x}_{t_j}$ as $\mathbf{x}_j$ and $\alpha(t_j)$ as α_j, $\gamma(t_j)$ as γ_j. Since $t_0 \approx 0, \mathbf{x}_0 \approx \mathbf{x}$, we could conclude the trained model $\boldsymbol{f}_{\boldsymbol{\theta}^*}(\mathbf{x}_1, t_1) = \mathbb{E}_{\mathbf{x}|\mathbf{x}_1}\left[\mathbf{x}|\mathbf{x}_1\right]$, and concequently

$$\boldsymbol{f}_{\boldsymbol{\theta}^*}(\mathbf{x}_{j+1}, t_{j+1}) = \mathbb{E}_{\mathbf{x}_j|\mathbf{x}_{j+1}}\left[\boldsymbol{f}_{\boldsymbol{\theta}^*}(\mathbf{x}_j, t_j)|\mathbf{x}_{j+1}\right], j = 1, \ldots, T-1 .$$

Using the property of the conditional expectation, we have $\mathbb{E}_{\mathbf{x}_j}\left[\boldsymbol{f}_{\boldsymbol{\theta}^*}(\mathbf{x}_j, t_j)\right] = \mathbb{E}_{\mathbf{x}}\left[\mathbf{x}\right], \forall j$. Now assume $\alpha^2(t) + \gamma^2(t) = 1, \forall t$. Using the expressions above we have

$$\boldsymbol{f}_{\boldsymbol{\theta}^*}(\mathbf{x}_1, t_1) = (1 - \gamma_1^2)\boldsymbol{\mu} + \gamma_1 \mathbf{x}_1$$

and

$$\boldsymbol{f}_{\boldsymbol{\theta}^*}(\mathbf{x}_j, t_j) = \boldsymbol{\mu} + (\gamma_1 \prod_{k=2}^{j} \beta_k) \cdot (\mathbf{x}_j - \gamma_j \boldsymbol{\mu}),\ j = 2, \ldots, T$$

where $\beta_k = \alpha_{k-1}\alpha_k + \gamma_{k-1}\gamma_k$ and $\gamma_1 \approx 1$. Further denote $c_j = \gamma_1 \prod_{k=2}^{j} \beta_k$ and assume $\alpha = \sin(t), \gamma(t) = \cos(t)$. For appropriate choice of the partition scheme (e.g. even or geometric), the coefficient c_j can converge as T grows. For instance, when evenly partitioning the interval $[0, t]$, we have:

$$\lim_{T\to\infty} c(t) = \lim_{T\to\infty} \gamma_1 \prod_{k=2}^{T} \beta_k = \lim_{T\to\infty} (\cos(\frac{t}{T}))^T = 1 .$$

Thus the trained model can be viewed as an interpolant between the consistency model($\lambda \to 1$ or $T \to \infty$) and the flow model($\lambda \to 0$ or $T \to 1$):

$$\boldsymbol{f}_{\boldsymbol{\theta}^*}(\mathbf{x}_t, t) = (1 - \gamma(t)c(t))\boldsymbol{\mu} + c(t)\mathbf{x}_t ,$$

$$\boldsymbol{f}_{\boldsymbol{\theta}^*}^{CM}(\mathbf{x}_t, t) = (1 - \gamma(t))\boldsymbol{\mu} + \mathbf{x}_t ,$$

$$\boldsymbol{f}_{\boldsymbol{\theta}^*}^{FM}(\mathbf{x}_t, t) = (1 - \gamma^2(t))\boldsymbol{\mu} + \gamma(t)\mathbf{x}_t .$$

The expression of $\boldsymbol{f}_{\boldsymbol{\theta}^*}^{CM}$ can be obtained by first compute the velocity field $\mathbf{v}^*(\mathbf{x}_t, t) = \mathbb{E}\left[\alpha'(t)\mathbf{z} + \gamma'(t)\mathbf{x}|\mathbf{x}_t\right] = \gamma'(t)\boldsymbol{\mu}$ then solve the initial value problem of ODE to get $\mathbf{x}(0)$.

The above optimal solution can be possibly obtained by training. For example if we set the parameterizition as $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = (1 - \gamma_t c_t)\boldsymbol{\theta} + c_t \mathbf{x}_t$, the gradient of the loss can be computed as (let $r = \lambda \cdot t$):

$$\nabla_{\boldsymbol{\theta}}\|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_r,r)\|_2^2=2(1-\gamma_t c_t)\left[(\alpha_t\gamma_t-\alpha_r\gamma_r)\mathbf{z}+(\gamma_r c_r-\gamma_t c_t)(\boldsymbol{\theta}-\mathbf{x})\right]\,,$$

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathbf{z},\mathbf{x}}\,\|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_r,r)\|_2^2=2(1-\gamma_t c_t)(\gamma_r c_r-\gamma_t c_t)(\boldsymbol{\theta}-\boldsymbol{\mu})\,,$$

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})&=\mathbb{E}_t\,\frac{2(1-\gamma_t c_t)(\gamma_r c_r-\gamma_t c_t)}{\hat{\omega}(t)}(\boldsymbol{\theta}-\boldsymbol{\mu})\\&=C(\boldsymbol{\theta}-\boldsymbol{\mu}),\,C=\mathbb{E}_t\,\frac{2(1-\gamma_t c_t)(\gamma_r c_r-\gamma_t c_t)}{\hat{\omega}(t)}\,.\end{aligned}$$

Use gradient descent to update $\boldsymbol{\theta}$ during training:

$$\frac{d\boldsymbol{\theta}(s)}{ds}=-\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})=-C(\boldsymbol{\theta}-\boldsymbol{\mu})\,.$$

The generalization loss thus evolves as:

$$\begin{aligned}\frac{d\|\boldsymbol{\theta}(s)-\boldsymbol{\mu}\|^2}{ds}&=\langle\boldsymbol{\theta}(s)-\boldsymbol{\mu},\frac{d\boldsymbol{\theta}(s)}{ds}\rangle\\&=\langle\boldsymbol{\theta}(s)-\boldsymbol{\mu},-C(\boldsymbol{\theta}(s)-\boldsymbol{\mu})\rangle\\&=-C\|\boldsymbol{\theta}(s)-\boldsymbol{\mu}\|^2\,,\\\implies\|\boldsymbol{\theta}(s)-\boldsymbol{\mu}\|^2&=\|\boldsymbol{\theta}(0)-\boldsymbol{\mu}\|^2e^{-Cs}\,.\end{aligned}$$

B.1.6 Unified Sampling Process

Deterministic sampling. When the stochastic ratio $\rho=0$, let's analyze a apecial case where the coefficients satisfying $\hat{\alpha}(t)=\frac{\mathrm{d}\alpha(t)}{\mathrm{d}t},\hat{\gamma}(t)=\frac{\mathrm{d}\gamma(t)}{\mathrm{d}t}$. Let $\Delta t=t_{i+1}-t_i$, for the core updating rule we have:

$$\begin{aligned}\mathbf{x}'&=\alpha(t_{i+1})\cdot\hat{\mathbf{z}}+\gamma(t_{i+1})\cdot\hat{\mathbf{x}}\\&=(\alpha(t_i)+\alpha'(t_i)\Delta t+o(\Delta t))\cdot\hat{\mathbf{z}}+(\gamma(t_i)+\gamma'(t_i)\Delta t+o(\Delta t))\cdot\hat{\mathbf{x}}\\&=(\alpha(t_i)\hat{\mathbf{z}}+\gamma(t_i)\hat{\mathbf{x}})+(\hat{\alpha}(t_i)\hat{\mathbf{z}}+\hat{\gamma}(t_i)\hat{\mathbf{x}})\cdot\Delta t+o(\Delta t)\\&=(\alpha(t_i)\boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F},\tilde{\mathbf{x}},t_i)+\gamma(t_i)\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F},\tilde{\mathbf{x}},t_i))+(\hat{\alpha}(t_i)\boldsymbol{f}^{\mathbf{z}}(\boldsymbol{F},\tilde{\mathbf{x}},t_i)+\hat{\gamma}(t_i)\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F},\tilde{\mathbf{x}},t_i))\cdot\Delta t+o(\Delta t)\\&=(\alpha(t_i)\frac{\hat{\gamma}(t_i)\cdot\tilde{\mathbf{x}}-\gamma(t_i)\cdot\boldsymbol{F}(\tilde{\mathbf{x}},t_i)}{\alpha(t_i)\cdot\hat{\gamma}(t_i)-\hat{\alpha}(t_i)\cdot\gamma(t_i)}+\gamma(t_i)\frac{\alpha(t_i)\cdot\boldsymbol{F}(\tilde{\mathbf{x}},t_i)-\hat{\alpha}(t_i)\cdot\mathbf{x}_t}{\alpha(t_i)\cdot\hat{\gamma}(t_i)-\hat{\alpha}(t_i)\cdot\gamma(t_i)})\\&+(\hat{\alpha}(t_i)\frac{\hat{\gamma}(t_i)\cdot\tilde{\mathbf{x}}-\gamma(t_i)\cdot\boldsymbol{F}(\tilde{\mathbf{x}},t_i)}{\alpha(t_i)\cdot\hat{\gamma}(t_i)-\hat{\alpha}(t_i)\cdot\gamma(t_i)}+\hat{\gamma}(t_i)\frac{\alpha(t_i)\cdot\boldsymbol{F}(\tilde{\mathbf{x}},t_i)-\hat{\alpha}(t_i)\cdot\mathbf{x}_t}{\alpha(t_i)\cdot\hat{\gamma}(t_i)-\hat{\alpha}(t_i)\cdot\gamma(t_i)})\cdot\Delta t+o(\Delta t)\\&=\tilde{\mathbf{x}}+\boldsymbol{F}(\tilde{\mathbf{x}},t_i)\cdot\Delta t+o(\Delta t)\end{aligned}$$

In this case $\boldsymbol{F}(\cdot,\cdot)$ tries to predict the velocity field of the flow model, and we can see that the term $\tilde{\mathbf{x}}+\boldsymbol{F}(\tilde{\mathbf{x}},t_i)\cdot\Delta t$ corresponds to the sampling rule of the Euler ODE solver.

Stochastic sampling. As for case when the stochastic ratio $\rho\neq0$, follow the Euler-Maruyama numerical methods of SDE, the noise injected should be a Gaussian with zero mean and variance proportional to Δt, so when the updating rule is $\mathbf{x}'=\alpha(t_{i+1})\cdot(\sqrt{1-\rho}\cdot\hat{\mathbf{z}}+\sqrt{\rho}\cdot\mathbf{z})+\gamma(t_{i+1})\cdot\hat{\mathbf{x}}$, the coefficient of $\mathbf{z}$ should satisfy

$$\alpha(t_{i+1})\sqrt{\rho}\propto\sqrt{\Delta t},\ \rho\propto\frac{\Delta t}{\alpha^2(t_{i+1})}$$

In practice, we set

$$\rho=\frac{2\Delta t\cdot\alpha(t_i)}{\alpha^2(t_{i+1})}\,.$$

which corresponds to $g(t)=\sqrt{2\alpha(t)}$ for the SDE $\mathrm{d}\mathbf{x}=\boldsymbol{f}(\mathbf{x},t)\mathrm{d}t+g(t)\mathrm{d}\boldsymbol{w}$.

B.2 Other Techniques

B.2.1 Beta Transformation

We utilize three representative cases to illustrate how the Beta transformation $f_{\text{Beta}}(t; \theta_1, \theta_2)$ generalizes time warping mechanisms for $t \in [0, 1]$.

Standard logit-normal time transformation [46, 7]. For $t \sim \mathcal{U}(0, 1)$, the logit-normal transformation $f_{\text{lognorm}}(t; 0, 1) = \frac{1}{1+\exp(-\Phi^{-1}(t))}$ generates a symmetric density profile peaked at $t = 0.5$, consistent with the central maximum of the logistic-normal distribution. Analogously, the Beta transformation $f_{\text{Beta}}(t; \theta_1, \theta_2)$ (with $\theta_1, \theta_2 > 1$) produces a density peak at $t = \frac{\theta_1 - 1}{\theta_1 + \theta_2 - 2}$. When $\theta_1 = \theta_2 > 1$, this reduces to $t = 0.5$, mirroring the logit-normal case. Both transformations concentrate sampling density around critical time regions, enabling importance sampling for accelerated training. Notably, this effect can be equivalently achieved by directly sampling $t \sim \text{Beta}(\theta_1, \theta_2)$.

Uniform time distribution [46, 48, 29, 25]. The uniform limit case emerges when $\theta_1 = \theta_2 = 1$, reducing $f_{\text{Beta}}(t; 1, 1)$ to an identity transformation. This corresponds to a flat density $p(t) = 1$, reflecting no temporal preference—a baseline configuration widely adopted in diffusion and flow-based models.

Approximately symmetrical time distribution [37, 36, 18, 20]. For near-symmetric configurations where $\theta_1 \approx \theta_2 > 1$, the Beta transformation induces quasi-symmetrical densities with tunable central sharpness. For instance, setting $\theta_1 = \theta_2 = 2$ yields a parabolic density peaking at $t = 0.5$, while $\theta_1 = \theta_2 \to 1^+$ asymptotically approaches uniformity. This flexibility allows practitioners to interpolate between uniform sampling and strongly peaked distributions, adapting to varying requirements for temporal resolution in training. Such approximate symmetry is particularly useful in consistency models where balanced gradient propagation across time steps is critical.